

AD-A141 405

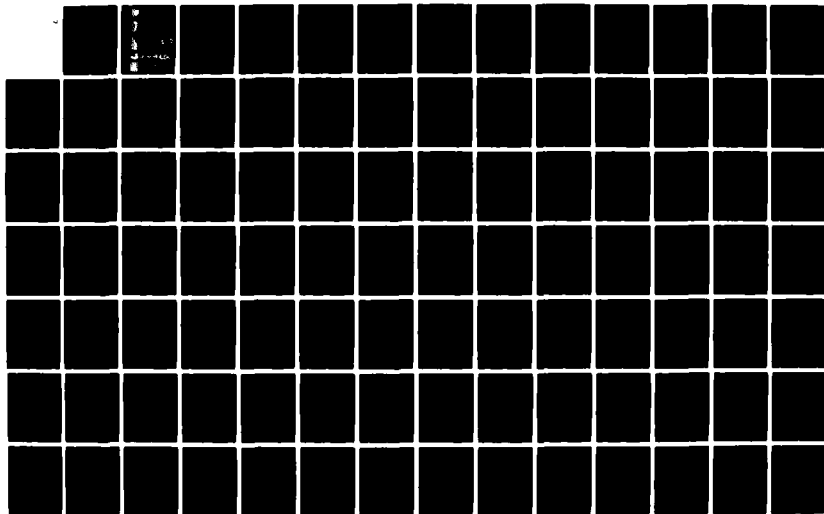
A REAL-TIME AIR DISPERSION MODELING SYSTEM(U) SIERRA
GEOPHYSICS INC REDMOND WA D E BLEEKER ET AL. APR 84
AFESC/ESL-TR-83-63 F08635-82-C-0374

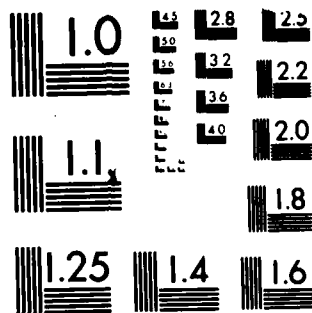
1/2

UNCLASSIFIED

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

ESL-TR-83-63

A Real-Time Air Dispersion Modeling System

D.E. BLEEKER, G. GARRABRANT and G.G. WORLEY
SIERRA GEOPHYSICS, INC
15446 BELL-RED ROAD, SUITE 400
REDMOND, WA 98052

APRIL 1984

FINAL REPORT
JULY 1982 - DECEMBER 1983

DTIC
ELECTE
MAY 23 1984
A

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

AD-A141 405

DTIC FILE COPY



ENGINEERING & SERVICES LABORATORY
AIR FORCE ENGINEERING & SERVICES CENTER
TYNDALL AIR FORCE BASE, FLORIDA 32403

84 05 22 022

NOTICE

PLEASE DO NOT REQUEST COPIES OF THIS REPORT FROM
HQ AFESC/RD (ENGINEERING AND SERVICES LABORATORY).

ADDITIONAL COPIES MAY BE PURCHASED FROM:

NATIONAL TECHNICAL INFORMATION SERVICE
5285 PORT ROYAL ROAD
SPRINGFIELD, VIRGINIA 22161

FEDERAL GOVERNMENT AGENCIES AND THEIR CONTRACTORS
REGISTERED WITH DEFENSE TECHNICAL INFORMATION CENTER
SHOULD DIRECT REQUESTS FOR COPIES OF THIS REPORT TO:

DEFENSE TECHNICAL INFORMATION CENTER
CAMERON STATION
ALEXANDRIA, VIRGINIA 22314

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ESL TR 83-63	2. GOVT ACCESSION NO. A141405	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Real-Time Air Dispersion Modeling System		5. TYPE OF REPORT & PERIOD COVERED Final Report July 1982 - December 1983
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Daniel E. Bleeker Gary Garrabrant Gary G. Worley		8. CONTRACT OR GRANT NUMBER(s) F08635-82-C-0374
9. PERFORMING ORGANIZATION NAME AND ADDRESS Sierra Geophysics, Inc. 15446 Bell-Red Road, Suite 400 Redmond, WA 98052		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 63723F 21039023
11. CONTROLLING OFFICE NAME AND ADDRESS HQ AFESC/RDVA Tyndall AFB FL 32403		12. REPORT DATE April 1984
		13. NUMBER OF PAGES 159
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES Availability of this report is shown on reverse of front cover.		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
Diffusion	Hazard Corridors	Safety
Dispersion	Evacuation Areas	Meteorology
Air Pollution	Emergency Actions	Toxic Hazards
Toxic Corridors	Spills of Toxic Chemicals	
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This report documents a microcomputer-based system for the real-time computation of toxic corridors associated with chemical releases. The program assists the user with interactive input of critical meteorological parameters and accesses user-specified data bases of information regarding toxic chemical attributes. Graphic displays show the analyst the resulting toxic corridor superimposed upon site-specific base maps. The program is documented for the user through help files. An archive mode provides a chronological history of all events, including the time and date of each calculation. The program is modular in		

DD FORM 1 JAN 73 1473

EDITION OF 1 NOV 68 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

OVER

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

design, allowing for modifications and upgrades in the future. ↗

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

PREFACE

This report was prepared by Sierra Geophysics, Inc., Redmond, Washington, under Contract F08635-82-C-0374, for the Air Force Engineering and Services Center, Environics Division (AFESC/RDV), Tyndall Air Force Base, Florida. Major Gary Worley was the AFESC Project Director; the Project Leader at Sierra Geophysics was Daniel Bleeker, assisted by Gary Garabrant.

The primary objective of the work carried out on this project between July 1982 and December 1983 was to place on an Air Force microcomputer the procedures that were heretofore performed manually to predict the hazard corridor associated with a toxic propellant release. Computer hardware specifications were established which allow for expanded dispersion modeling techniques, and follow-on efforts are underway to upgrade the software contained in appendices to this report.

The use of certain computer equipment in this project does not constitute an endorsement of these products by the Air Force, and the views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policy, either expressed or implied, of the Air Force or the United States Government.

This report has been reviewed by the Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be available to the general public, including foreign nationals.

This technical report has been reviewed and is approved for publication.

Gary G. Worley

GARY G. WORLEY, Maj, USAF
Project Officer

Jimmy N. Fulford

JIMMY N. FULFORD, Lt Col, USAF
Chief, Environics Division

Robert E. Boyer

ROBERT E. BOYER, Col, USAF
Director, Engineering and Services
Laboratory



Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Avail and/or	
Spec. S. Avail	
A1	

TABLE OF CONTENTS

SECTION	TITLE	PAGE
I	INTRODUCTION.....	1
II	DESCRIPTION OF THE RTMS.....	4
III	IMPLEMENTATION OF THE RTMS.....	9
IV	CONCLUSIONS.....	13
V	REFERENCES.....	14
APPENDIX		
A	DATA BASE DESCRIPTION.....	17
B	PROGRAM LISTINGS.....	23

LIST OF FIGURES

FIGURE	TITLE	PAGE
1	TOXIC CORRIDOR DEVELOPMENT.....	2
2	TOXIC CORRIDOR INFORMATION DISPLAY..	3
3	FUNCTIONAL STRUCTURE OF THE RTMS....	5
4	RTMS COMPUTER SYSTEM.....	10
5	GRAPHICAL OUTPUT DISPLAY.....	13
A1	PROCEDURE DATA BASE FILE STRUCTURE..	18
A2	DBMM AND THE SUBSTANCE-SOURCE DATA BASE.....	19
A3	SUBSTANCE-SOURCE DATA BASE STRUCTURES.....	21

LIST OF TABLES

TABLE	TITLE	PAGE
1	RTMS SYSTEM SPECIFICATIONS.....	11

SECTION I

INTRODUCTION

When a toxic chemical is vented to the atmosphere, an estimate of the extent of the affected area, or "toxic corridor" must be made. The size of the toxic corridor depends upon the release rate of the toxic chemical, the concentration exposure limits for the chemical and the dispersive properties of the atmosphere at the time of release. Four methods of producing toxic corridor estimates, ranging from table look-up to programmable calculator techniques, have been described in previous work (Reference 1). This report describes a fifth approach: the use of a microcomputer system which encompasses crucial support functions, in addition to toxic corridor computations, and thereby expedites overall emergency response.

The primary goal of the computerized Real-Time Modeling System (RTMS) is to separate the tasks suitable for execution on a computer system from those which must be performed by an emergency response specialist. Figure 1 presents the basic tasks involved in toxic corridor calculations. The RTMS consolidates the data manipulation, toxic corridor computations, and plotting functions, allowing more time for subjective analysis of the results of the toxic corridor calculations.

The RTMS consolidates information pertinent to a toxic chemical release, as shown by Figure 2. Critical meteorological parameters and information about the toxic chemical are simultaneously displayed, aiding user analysis. The flexible nature of the RTMS allows inclusion of additional information in the graphic display to meet special requirements.

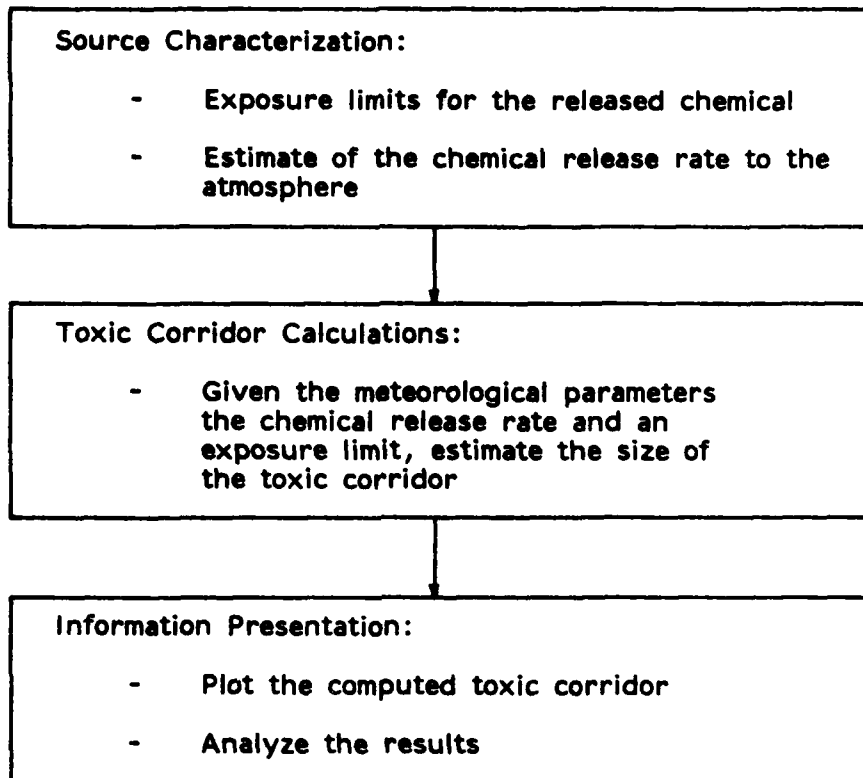


Figure 1. Toxic Corridor Development

TOXIC CORRIDOR INFORMATION

TIME: 17:36:15
DATE: 10/19/83

SUBSTANCE: OXIDIZER
SOURCE : PRESSURE DRAIN FIXED SYSTEM STG I&II
SOURCE STRENGTH (#/MIN): 5.00

METEOROLOGICAL DATA

WIND SPEED..... (KTS): 20.0
WIND DIRECTION. (DEG): 270.0
SIGMA THETA.... (DEG): 12.0
DELTA T..... (F): -1.0

CORRIDOR DIRECTION. (DEG): 90.0
CORRIDOR WIDTH..... (DEG): 72.0

SPEL	(PPM)	CORRIDOR LENGTH (FEET)	
		OB/DG (DELTA T)	OB/DG (SIGMA THETA)
10 MIN	5.0	777.6	719.3
30 MIN	3.0	1010.5	933.4
60 MIN	2.0	1244.2	1147.8

Figure 2. Toxic Corridor Information Display

SECTION II

DESCRIPTION OF THE RTMS

The following sections describe the function of each major module of the RTMS. Figure 3 represents a block diagram showing the interrelationship of each module.

A. COMMAND AND CONTROL MODULE (CCM)

The CCM is the primary user interface with the RTMS. Each of the six functional areas shown in Figure 3 is accessed via a menu presented by the CCM.

B. EVENT ARCHIVE MODE

When the Event Archive Mode is selected, the results of all subsequent toxic corridor calculations are stored on the system in an archive file. The Event Archive mode provides a chronological history of all events, including the time and date of each calculation.

C. RECALL EVENT ARCHIVE

The Recall Event Archive utility provides the user with a list of stored event archive data. Users may select any of the stored event data for display and/or printing.

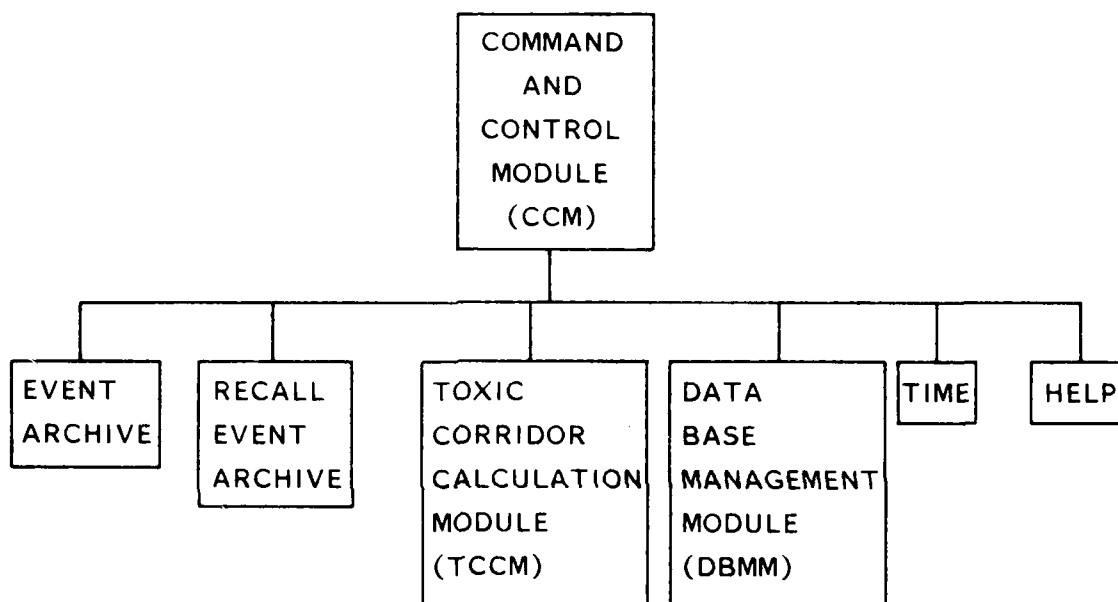


Figure 3. Functional Structure of the RTMS

D. TOXIC CORRIDOR CALCULATION MODULE (TCCM)

Toxic corridor lengths are calculated, based on the Ocean Breeze/
Dry Gulch equation (Reference 1):

$$X = P \left[3.28 \left(\frac{29.75}{\text{GMW}} \right)^{0.513} \left(\frac{C_p}{Q} \right)^{-0.513} \left(\Delta T + 10 \right)^{2.53} \right]$$

where X = downwind distance from the source in feet. As used here, this distance defines the toxic corridor length.

P = a probability factor based upon the probability that a specified concentration is not exceeded outside the corridor. Calculations in Kahler and others (Reference 1) assume a 90 percent probability with P equal to 1.63.

GMW = gram molecular weight of the toxic chemical.

C_p = peak concentration in parts per million by volume (PPM) along a plume centerline and at a height of approximately 5 feet above the ground at a given downwind travel distance, X, in feet. Toxic corridor lengths are calculated by using a specified exposure limit for C_p in the above equation.

Q = source strength in lb/min.

ΔT = the temperature in °F at 54 feet minus the temperature at 6 feet (NOTE: A negative ΔT means a decrease of temperature with height and a positive ΔT means an increase with height.)

Values for GMW, CP, and Q may be obtained from the RTMS data base or entered by the user. Three values are used for peak concentration, corresponding to the 10-, 30- and 60-minute exposures of a given toxic substance. A value for ΔT must be supplied by the user. Validity checks are performed on the data, then corridor lengths for appropriate concentration levels are calculated.

If the wind speed is greater than or equal to 3 knots, the corridor direction is determined from the wind direction entered by the user. The corridor width is set to $6 \sigma \theta$ ($\sigma \theta$ is the standard deviation of the horizontal wind direction). The width is set to 360 degrees if the wind speed is less than 3 knots.

A simplified algorithm for computing source strengths due to toxic chemical spills (Reference 2) is available under the TCCM. If the spill option is selected, the user enters information concerning the size of the spill (square feet of area covered) and the temperature of the pool. A source strength is calculated and the TCCM computes corridor width and lengths. Figure 2 shows the typical output from the TCCM.

E. DATA BASE MANAGEMENT MODULE (DBMM)

Two distinct types of information are maintained by the DBMM: (1) Source-substance data and (2) Procedural information. Source-substance data include specific values for source and substance parameters necessary for toxic corridor calculations. A full description of the source-substance data base is available in Appendix A. The procedural data are of a documentary nature, providing the RTMS user access to background information while executing the RTMS. For example, the user may, while running the RTMS program, display detailed information concerning the Ocean Breeze/Dry Gulch equation.

The DBMM provides the user with the means of searching, modifying, adding to, and deleting from the source-substance and procedural data bases.

F. TIME

The "time" utility allows the user to display current time, as determined by the computer system which is host to the RTMS.

G. HELP

The entire RTMS program is self-documenting. The "help" utility presents the user with a description of each of the options presented in the selection menu.

SECTION III

IMPLEMENTATION OF THE RTMS

The RTMS, as described here, is implemented on a Cromemco 68000 computer system, operating under the CROMIX multiuser operating system. The basic components of the computer system are shown in Figure 4 and detailed specifications are listed in Table 1.

The RTMS is programmed entirely in FORTRAN 77 (as implemented under the CROMIX operating system). For details on the specifics of the FORTRAN 77, one should consult the Cromemco 68000 FORTRAN 77 manual. Complete program listings for the RTMS are available in Appendix B.

A. STRUCTURE OF THE RTMS PROGRAM

The RTMS is composed of two types of program modules: (1) high-level routines designed for performing the primary tasks of data base management, toxic corridor calculation, etc., and (2) low-level routines for menu display and command processing. The high-level routines are functionally grouped to perform the primary tasks described in Section II of this report. The low-level routines are used as required for display and command processing by any of the high-level routines.

The demarkation between high- and low-level routines greatly facilitates transporting the RTMS software between computer systems. The RTMS software may be fully implemented on a host computer system, given adequate random access memory and a FORTRAN 77 compiler, with relatively minor changes to the low-level routines.

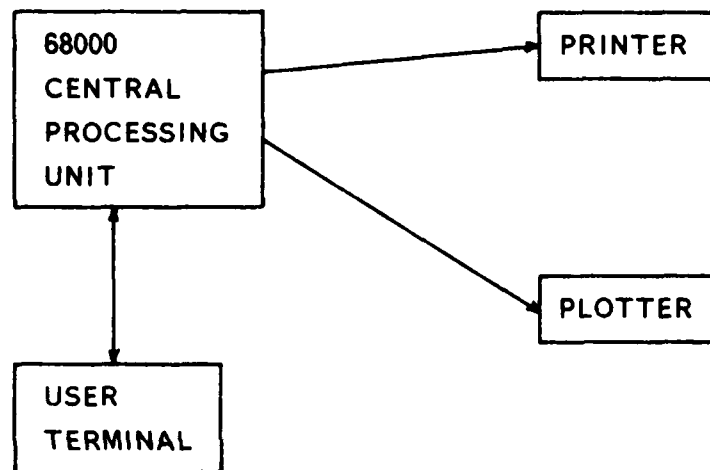


Figure 4. RTMS Computer System

TABLE 1
RTMS SYSTEM SPECIFICATIONS

1. Computer

- Cromemco 68000/Z80 Computer
- CROMIX Operating System
- 512 Kilobytes of Random Access Memory (RAM)
- 2 - 5.25-Inch Floppy Disk Drives
- 1 - 20-Million Byte Rigid Disk Drive
- FORTRAN 77 Software
- 3 - RS232 Serial I/O Ports
- 1 - Parallel Port (for the printer)

2. Printer

- 132-Column Dot Matrix Printer with a Parallel Interface

3. Plotter

- 8-Pen Plotter with a Serial Interface

B. OPERATION OF THE RTMS

The RTMS program is self-documented, via "help" files. The "help" information is intended to aid the inexperienced person in effectively using the RTMS. Help is available to the user as a menu option and is therefore available during the operation of RTMS.

C. RTMS OUTPUT

Several types of output are available from the current version of RTMS. A tabular display (shown previously in Figure 2) contains a summary of the input data, output data and other pertinent information from the source-substance data base that was used in the calculation. This summary can be directed from the CRT to a printer, or it can be routed to a communications terminal.

A graphical display, as shown in Figure 5, is directed to the plotter. The hazard corridor can be plotted at a specified location on a digitized map from the data base, or the plot can serve as an overlay onto any base map of an appropriate scale.

CORRIDOR INFORMATION
 OCEAN BREEZE - DRY GULCH EQUATION
 BASED ONLY ON DELTA T)

DIRECTION: 184.8°
 WIDTH : 30.8°

10	:	10 MIN SPEL
30	:	30 MIN SPEL
60	:	60 MIN SPEL
PZ	:	PRIORITY ZONE (10000 FEET)

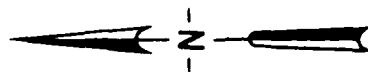
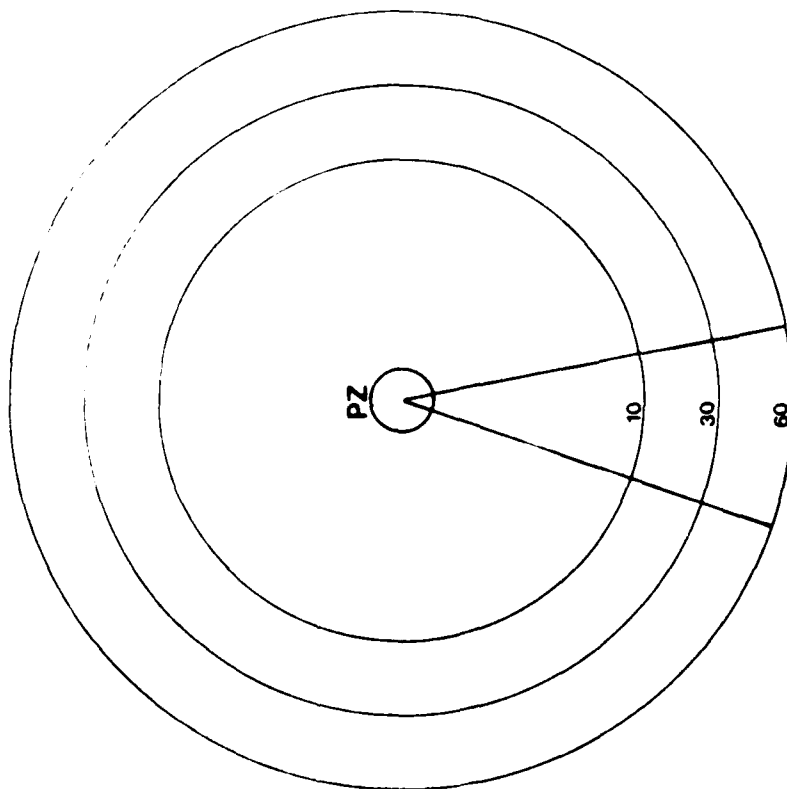


Figure 5. Graphical Output Display (original in color).

SECTION IV

CONCLUSIONS

The RTMS fulfills its primary requirement of streamlining the toxic corridor calculations requirement by consolidating the data required to perform the corridor computations. Additional benefits derive from the archive mode of the RTMS, which provides automatic record keeping of all corridor determinations, and from the maintenance of source-substance and procedural data bases. The RTMS is fully menu-driven and provides on-line "help" information, facilitating system use by those unfamiliar with its operation.

A valuable enhancement to the existing RTMS will be the inclusion of interactive color graphics capabilities. The graphics capabilities will allow users to interact with the RTMS via displayed maps and pictorial representations of the data bases.

The RTMS, as developed, is oriented towards supporting TITAN II operations, but provides the framework for a much broader application. The data base capability is being extended to support a complete inventory of toxic substances for an arbitrary location, and the toxic corridor calculation module is being enhanced with a more advanced air dispersion model. Additional capabilities could be added, ultimately resulting in a fully integrated emergency response and toxic substance record-keeping system.

SECTION V

REFERENCES

1. Kahler, J..P., R.G. Curry, and R.A. Kandler, Calculating Toxic Corridors, Headquarters Air Weather Service (MAC), Scott AFB, IL, AWSTR-80-003, 1980.
2. Clewell, H.J., A Simple Formula for Estimating Source Strengths from Spills of Toxic Liquids, ESL TR 83-03, Engineering and Services Laboratory, Air Force Engineering and Services Center, Tyndall AFB, Florida, 1983.
3. Haugen, D.A., and J.H. Taylor (Editors), The Ocean Breeze and Dry Gulch Diffusion Programs - Volume II, AFCRL - 63 - 791 (2), December 1963.

APPENDIX A

A. PROCEDURE DATA BASE

The procedure data base consists of an unformatted direct access file with a logical record length of 48 bytes. This file is called the procedure header file (PH file). The first record in this file contains a 4-byte integer which indicates how many records are in the file. The remainder of the records take the form shown in Figure A1.

The 40-character procedure name or description field is used as a key to identify each procedure. The actual name of the disk file in which the text for a procedure is contained is in the last eight characters of the record. These file names take the form PROCXXX where XXX ranges between 000 and 099.

The PROCXXX files are sequential formatted files whose logical record length is 80 bytes. These files contain the actual text about any procedure which is on file.

B. SUBSTANCE-SOURCE DATA BASE

The Substance-Source Data base is made up of four direct access unformatted files. The four file types are: Substance, Source, Pointer and Data. The relationship between the DBMM and the Substance-Source Data base is shown by Figure A2.

PROCEDURE FILE RECORD

PROCEDURE NAME OR DESCRIPTION	PROCEDURE FILE CONTAINING INFO PROCXXX
40 CHARACTERS	8 CHARACTERS

Figure A1. Procedure Data Base File Structure.

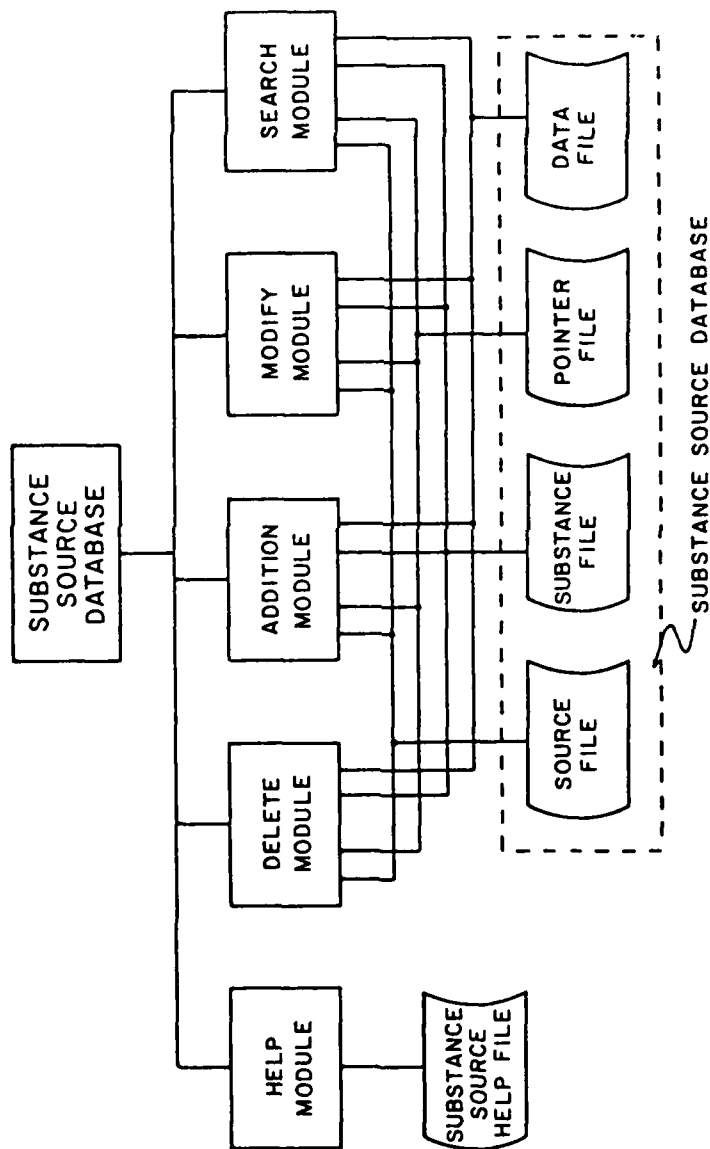


Figure A2. DBMM and the Substance-Source Data Base

The record length for the Source File is 40 bytes. The first record is a header record and contains a counter that indicates how many records are in the file. The counter is a 4-byte integer. The rest of the records take the form shown in Figure A3. The source name is stored in the file as indicated in Figure A3.

The Substance File has a logical record length of 120 bytes. The first 40 bytes are used to store the substance name. Since many of the quantities of interest are substance-related, they are stored in the substance record. The Substance File also has a header record which holds a 4-byte integer counter to indicate the number of records in the file, including the header record. The last 40 bytes of the record hold ten 4-byte real numbers. They are the gram molecular weight (GMW), the 10-, 30- and 60-minute public emergency limits, the Z factor and five extra locations for future expansion. All of these data are part of the substance records as shown in Figure A3. The middle 40 bytes contain ten 4-byte integers. The first 9 are pointers into the source file and the 10th pointer points into the substance file. Each substance record is allowed to point to 9 sources. The first 9 pointers are record numbers into the source file. These pointers indicate which sources the substance is linked to. Since a substance may be linked to more than nine sources, there may be more than one substance (of a particular kind) record in the substance file. The 10th pointer points to the next record in the substance file for the same substance (or the pointer has a zero value). This allows the substances to be forward-linked in the substance file.

The Pointer File is easy to explain, but a little harder to understand. The Pointer File record length is 40 bytes. It does not contain a header record. However, for uniformity with the other files the first record must be skipped. The rest of the records have the form shown on Figure A3. Each record consists of ten 4-byte integers. The last integer is extra, only the first 9 are used. The pointers point into the data file. The *i*th record and *i*th pointer in that record correspond to the *i*th substance and *i*th source pointer in the substance

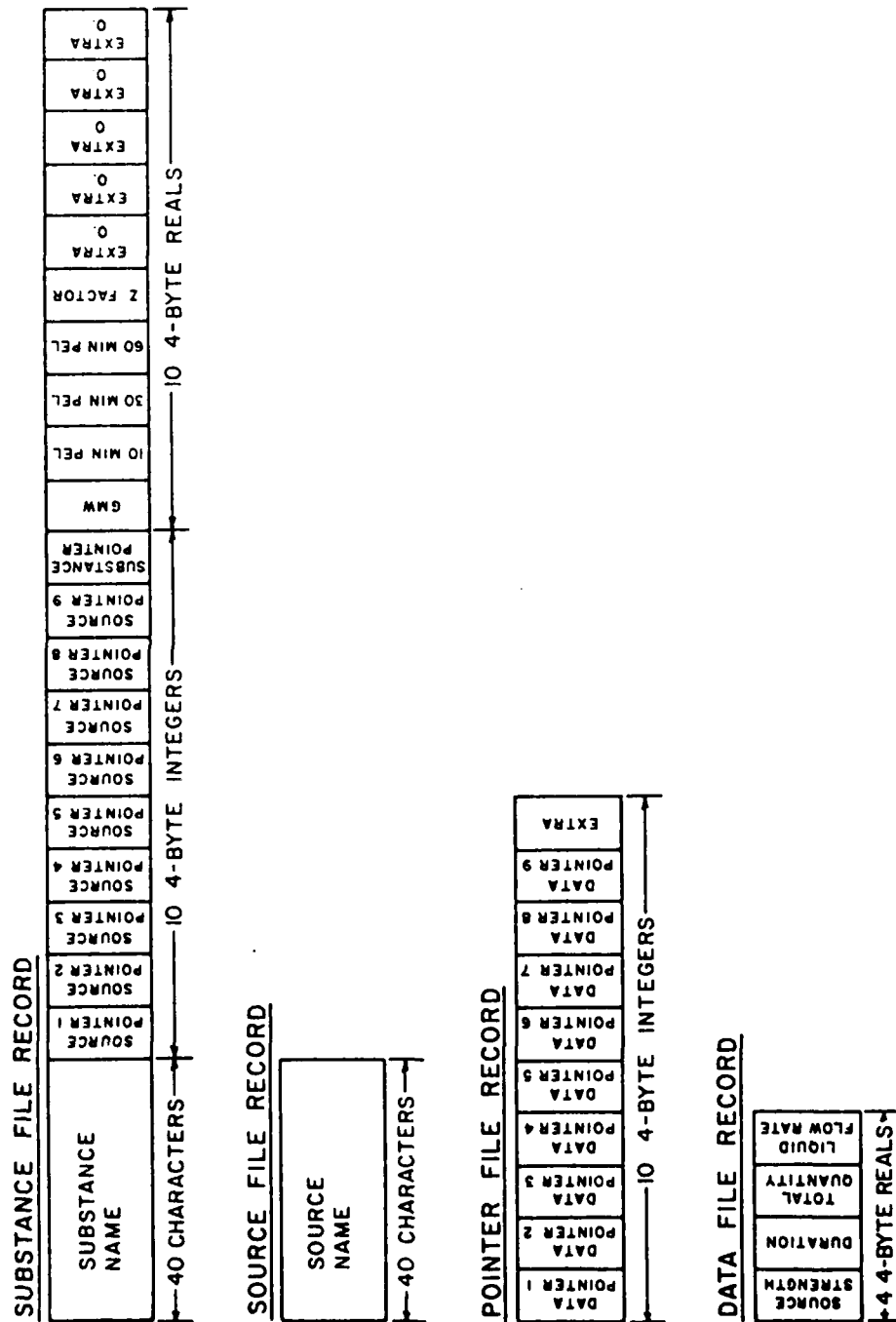


Figure A3. Substance-Source Data Base Structures.

record of interest. The pointer in the pointer record points to the record number in the data file that contains the source strength, duration, total quantity and liquid flow rate for a particular substance-source pair.

The Data File logical record length is 16 bytes. As before, the data file contains a header record. The header record contains a 4-byte integer counter which indicates the number of records in the file. The data stored in each record are source strength, duration, total quantity and liquid flow rate for a particular substance-source. The word locations are shown in Figure A3.

APPENDIX B

PROGRAM LISTINGS


```
$BIGCODE
PROGRAM CCM
```

```
C*****
C THIS IS THE MAIN DRIVING PROGRAM FOR TOXIC CORRIDOR
C CALCULATIONS. THIS PROGRAM PRODUCES A MENU TO SELECT
C WHICH OF THE MODULES, TCCM, DBCM, ... IS TO BE EXECUTED.
C !!! WARNING !!! DO NOT USE UNIT 15 IN SYSPARM FILE, IT IS RESERVED
C FOR THE PRINTER
C
C VARIABLES USED
C ITR - INTERACTIVE TERMINAL READ UNIT
C AUNIT - UNIT FOR ARCHIVE FILES
C AUNIT1 - UNIT FOR RECALLED ARCHIVE FILES
C HFILE - SYSTEM HELP FILE NAME
C HUNIT - UNIT FOR SYSTEM HELP FILE
C SFILE1 - SUBSTANCE FILE NAME (SUBSTANCE-SOURCE DATA BASE)
C SFILE2 - SOURCE " " "
C SFILE3 - POINTER " " "
C SFILE4 - SOURCE DATA FILE NAME "
C SUNIT1 - UNIT FOR SFILE1
C SUNIT2 - " " SFILE2
C SUNIT3 - " " SFILE3
C SUNIT4 - " " SFILE4
C SHFILE - SUBSTANCE-SOURCE HELP FILE NAME
C PFILE1 - PROCEDURE HEADER FILE NAME (PROCEDURE DATA BASE)
C PUNIT1 - UNIT FOR PFILE1
C PHFILE - PROCEDURE HELP FILE NAME
C SMFILE - MENU FILE NAME
C SMUNIT - UNIT FOR SMFILE
C BFILE - CROMIX DIRECTORY WHERE ARCHIVE FILES ARE STORED
C BUNIT - UNIT FOR OPENING CROMIX FILE DIRECTORY LIST. USED
C TO ESTABLISH A LIST OF AVAILABLE EVENT ARCHIVE FILES
C MFILE - MAP DATA BASE HEADER FILE NAME
C MUNIT - UNIT FOR MFILE
C*****
C
```

```
CHARACTER*40 BFILE
CHARACTER*1 CMD(1)
CHARACTER*15 STAMP
CHARACTER*7 AFILE, CFILE, SFILE1, SFILE2, SFILE3, SFILE4, MFILE,
* HFILE, SHFILE, PFILE1, PHFILE, SMFILE

INTEGER ITR, AUNIT, AUNIT1, HUNIT, SUNIT1, SUNIT2, SUNIT3, SUNIT4,
* CUNIT, PUNIT1, PUNIT2, SMUNIT, RC(2,3), BUNIT, MUNIT

LOGICAL AFLAG, MREAD

COMMON/MENUS/MREAD
```

```
C CHECK FOR THE SYSTEM FILE SYSPARM. IF IT DOESN'T EXIST THEN
C EXIT THE PROGRAM.
INQUIRE (FILE='SYSPARM', EXIST=AFLAG)
IF (.NOT. AFLAG) THEN
CALL CLEAR(7,0)
CALL MENU DR('SYSTEM FILE NOT FOUND', 12, 30, 2, 0, 1, 1)
CALL MENU DR(' ', 24, 1, 2, 0, 1, 1)
STOP
ENDIF
```

```

C      USER INPUT VALID, CALL ROUTINE TO DISPLAY THE FILE
      CALL PROREV(ITR,TEMP,AUNIT)

C      GO BACK AND DISPLAY THE MAIN MENU
      GOTO 100

      ENDIF

C      INCREMENT THE COUNTER WHICH TELLS HOW MANY ARCHIVE FILES HAVE BEEN FOUND
      CNT=CNT+1
      GOTO 20

C      END OF FILE REACHED IN ARFILE
21     IF ((FLAG1).AND.(.NOT. FLAG2)) THEN
          CLOSE (BUNIT)
          GOTO 15
      ENDIF

      IF ((FLAG1).AND.(FLAG2))
*      CALL MESS(17,RC(20,1),RC(20,2),RC(20,3),7)

C      THERE WERE NOT 19 FILES TO DISPLAY, ONLY CNT-1 TO DISPLAY
      CALL MENUWR(RC,20,2,CNT-1,OUT,0,2,ST)

C      INPUT THE USER OPTION
35     INP(1)= ' '
      CALL MENURD(RC,20,1,1,INP,ITR)

C      USER SELECTED TO RETURN, SO CLOSE THE LISTING FILE
      IF (INP(1).EQ. 'X ') THEN
          CLOSE (BUNIT)
          RETURN
      ENDIF

C      USER CHOOSE TO CONTINUE
      IF (INP(1).EQ. 'C ') THEN
          CLOSE (BUNIT)
          FLAG1=.TRUE.
          CALL MENUSV(SMFILE,115,RC,20,SMUNIT)
          GOTO 15
      ENDIF

C      USER SELECTED A NUMBER, MAYBE. CHECK TO SEE IF VALID INPUT AND TRY
C      TO OPEN THE ARCHIVE FILE
      READ(INP,'(BN,I3)',ERR=35) JJ

C      FORM THE ARCHIVE FILE NAME INCLUDING THE DIRECTORY PREFIX
      AFILE=EFILE(JJ)
      FNAME(1)(1:47)=BFILE
      FNAME(1)(41:47)=AFILE
      TEMP=FNAME(1)
      CALL PACK(TEMP,J)
      INQUIRE (FILE=TEMP,EXIST=FLAG3)
      IF (.NOT. FLAG3) GOTO 35

C      VALID USER INPUT, DISPLAY THE ARCHIVE FILE
      CALL PROREV(ITR,TEMP,AUNIT)
      CLOSE (BUNIT)

C      GO BACK AND DISPLAY THE MAIN MENU
      GOTO 100

```

```

END
SUBROUTINE PROREV(ITR, AFILE, AUNIT)

C*****
C    PROREV DISPLAYS THE LIST OF AVAILABLE ARCHIVE FILES ON THE SCREEN
C    22 LINES AT A TIME
C
C    VARIABLES ITR, AFILE AND AUNIT ARE DESCRIBED IN SUBROUTINE REVENT
C
C    INTERNAL FLAGS:
C        FLAG1    -   TRUE :   THE SCREEN SHOULD BE CLEARED
C                   FALSE:   THE SCREEN SHOULD NOT BE CLEARED
C        FLAG2    -   TRUE :   END OF FILE REACHED
C                   FALSE:   NO END OF FILE REACHED
C*****

CHARACTER*80  DLINE
CHARACTER*47  AFILE
CHARACTER*1   INP(1), FMFEED

INTEGER       AUNIT, RC(1, 3)

LOGICAL       FLAG1, FLAG2

RC(1, 1)=23
RC(1, 2)=34
RC(1, 3)=1

C    CLEAR THE SCREEN AND OPEN THE ARCHIVE FILE TO BE DISPLAYED
CALL CLEAR(7, 0)
OPEN (AUNIT, FILE=AFILE, STATUS='OLD')

C    STRIP OFF THE HEADER: ARCHIVE FILE ARCHXX   HH:MM:SS   MM/DD/YY
READ(AUNIT, '(A80)') DLINE

30  FLAG2=.FALSE.
    FLAG1=.TRUE.
    DO 10 I=1, 22, 1
        READ(AUNIT, '(A80)', END=14) DLINE
        IF (FLAG1) THEN
            FLAG1=.FALSE.
            CALL CLEAR(7, 0)
            ENDIF
        CALL MENU DR(DLINE, I, 1, 2, 0, 1, 1)
10  CONTINUE
    GOTO 15
14  FLAG2=.TRUE.

C    IF END OF FILE REACHED DISPLAY MESSAGE TO THAT AFFECT
15  IF (FLAG2) CALL MENU DR('END OF FILE REACHED', 23, 62, 7, 0, 1, 1)
    CALL MENU DR('SELECT OPTION (X OR C OR P) ==>', 23, 1, 2, 0, 1, 1)

C    INPUT THE USER SELECTED OPTION
20  INP(1)=' '
    CALL MENU DR(RC, 1, 1, 1, INP, ITR)

C    USER SELECTED THE RETURN OPTION, SO CLOSE FILE AND RETURN
    IF (INP(1) .EQ. 'X') THEN
        CLOSE (AUNIT)

```

```

                                RETURN
                                ENDIF

C   USER SELECTED THE PRINT OPTION, SO PRINT THE FILE
    IF (INP(1) .EQ. 'P') THEN
        CLOSE(AUNIT)
        OPEN(AUNIT, FILE=AFILE, STATUS='OLD')

C   SET THE PRINTER UP WITH FORM FEED
        IFORM= 12
        FMFEED= CHAR(IFORM)
        OPEN(15, FILE='/DEV/PRT')

C   STRIP OFF THE HEADER RECORD
        READ(AUNIT, '(A80)', END=220) DLINE
200    CONTINUE

C   SEND FORM FEED
        WRITE(15, '(A1)') FMFEED
        DO 210 I=1, 22
            READ(AUNIT, '(A80)', END=220) DLINE
            WRITE(15, '(A80)') DLINE
210    CONTINUE

        GO TO 200

C   SEND FORM FEED TO CLEAR PAGE, CLOSE UNITS
220    WRITE(15, '(A1)') FMFEED
        CLOSE(AUNIT)
        CLOSE(15)
        RETURN
        ENDIF

C   USER SELECTED TO CONTINUE VIEWING ARCHIVE FILE
    IF (INP(1) .EQ. 'C') THEN
        IF (FLAG2) THEN
            CLOSE (AUNIT)
            OPEN (AUNIT, FILE=AFILE, STATUS='OLD')
            READ(AUNIT, '(A80)') DLINE
            FLAG2=.FALSE.
            ENDIF
            GOTO 30
        ENDIF

        GOTO 20
    END
    INTEGER FUNCTION BNDX(STR, N)

C*****
C   THIS FUNCTION SCANS A CHARACTER STRING FROM POSITION N TO 1 AND RETURNS
C   THE FIRST NON-BLANK POSITION ENCOUNTERED, IF NO BLANKS ARE FOUND A 1 IS
C   RETURNED.
C   VARIABLES PASSED:
C
C   STR      - CHARACTER STRING TO BE SEARCHED
C   N        - POSITION OF THE STRING TO SEARCH BACKWARD FROM
C
C   VARIABLES RETURNED:
C
C   BNDX     - FIRST NON-BLANK POSITION ENCOUNTERED IN THE BACKWARD SEARCH
C*****

```

```

RETURN
END
CHARACTER*7 FUNCTION EFILE(I)

```

```

C*****
C    THIS FUNCTION FORMS THE FILE NAME ARCHX, WHERE X IS DETERMINED BY THE
C    VALUE OF I PASSED TO THE FUNCTION.
C    VARIABLES PASSED:
C
C    I - VALUE TO BE CONCATENATED ON ARCH
C
C    VARIABLES RETURNED:
C
C    EFILE - CHARACTER NAME ARCHX, X PASSED THROUGH THE WINDOW
C*****

```

```

CHARACTER*2 TEMP

```

```

INTEGER I

```

```

C    BUILD THE FILE NAME ARCHX, X=0,1,2,...,99
C    EFILE(1:7)='ARCH '
C    WRITE(TEMP(1:2),'(I2)') I
C    IF (TEMP(1:1) .EQ. ' ') THEN
C        EFILE(5:5)=TEMP(2:2)
C    ELSE
C        EFILE(5:6)=TEMP(1:2)
C    ENDIF

```

```

RETURN
END
SUBROUTINE PACK(STR,J)

```

```

C*****
C    THIS SUBROUTINE REMOVES THE BLANKS FROM WITHIN A CHARACTER STRING
C    AND PADS THE STRING WITH BLANKS
C
C    VARIABLES PASSED
C    STR - CHARACTER STRING TO BE PACKED
C
C    VARIABLES RETURNED
C    J - THE NUMBER OF NON-BLANK CHARACTERS IN THE RETURNED STRING
C*****

```

```

CHARACTER*(*) STR

```

```

INTEGER I,J

```

```

L=LEN(STR)
J=0

```

```

C    REMOVE THE BLANKS FROM THE CHARACTER STRING
C    DO 5 I=1,L,1
C        IF (STR(I:I) .NE. ' ') THEN
C            J=J+1
C            STR(J:J)=STR(I:I)
C        ENDIF

```

```

5    CONTINUE

```

```

C    PAD THE REST OF THE STRING WITH BLANKS

```

```

DO 10 I=J+1,L,1
  STR(I:I)=' '
10 CONTINUE

RETURN
END
SUBROUTINE REVENT(ITR,AUNIT,SMUNIT,SMFILE,BUNIT,BFILE)

```

```

C*****
C      REVENT RECALLS THE EVENT ARCHIVE FILES.  A LIST IS DISPLAYED ON THE
C      SCREEN ALLOWING THE USER TO SELECT THE DESIRED ARCHIVE FILE.  REVENT
C      ASSUMES THAT A DIRECTORY LISTING OF AVAILABLE ARCHIVE FILES HAS
C      BEEN CREATED BY THE HOST COMPUTER SYSTEM PRIOR TO RTMS EXECUTION.
C      THE DIRECTORY LISTING FILE MUST BE NAMED 'ARFILES' AND BE IN THE
C      DIRECTORY INDICATED BY ARGUMENT BFILE.

```

```

C      VARIABLES PASSED

```

```

C      ITR      -  INTERACTIVE TERMINAL READ UNIT
C      AUNIT    -  UNIT FOR RECALLED ARCHIVE FILE
C      SMFILE   -  MENU FILE NAME
C      SMUNIT   -  UNIT FOR SMFILE
C      BFILE    -  CROMIX DIRECTORY CONTAINING 'ARFILES'
C      BUNIT    -  UNIT FOR 'ARFILES'

```

```

C*****

```

```

CHARACTER*80 OUT(19),TLINE
CHARACTER*47 FNAME(1),FNAME1(1),TEMP,TEMP1
CHARACTER*40 BFILE
CHARACTER*7  AFILE,EFILE,TFILE,SMFILE
CHARACTER*3  INP(1)

```

```

LOGICAL      FLAG1,FLAG2,FLAG3

```

```

INTEGER      ITR,AUNIT,JJ,CNT,TUNIT,SMUNIT,RC(20,3),
*            ST(3),BUNIT

```

```

DATA ST/0,0,0/

```

```

C      FORM THE LISTING FILE NAME, PACK IT AND SEE IF IT EXISTS
      FNAME1(1)(1:47)=BFILE
      FNAME1(1)(41:47)='ARFILES'
      TEMP1=FNAME1(1)
      CALL PACK(TEMP1,J)
      INQUIRE (FILE=TEMP1,EXIST=FLAG1)
      IF (.NOT. FLAG1) RETURN

```

```

C      DISPLAY THE MAIN MENU
100 CALL MENUSV(SMFILE,115,RC,20,SMUNIT)

```

```

C      FLAG1 - INDICATES WHETHER THE SCREEN IS CLEAR
C      FLAG2 - INDICATES WHETHER ANY FILES EXIST
      FLAG1=.TRUE.
      FLAG2=.TRUE.

```

```

15  CNT=2
C      OPEN THE LISTING FILE
      OPEN (BUNIT,FILE=TEMP1,STATUS='OLD')

C      LOOK FOR ARCH FILES IN THE MASTER FILE

```

```

20  READ(BUNIT,'(A80)',END=21) TLINE
    IF (TLINE(18:21) .NE. 'arch') GOTO 20

C   FORM THE ARCHIVE FILE NAME INCLUDING THE DIRECTORY PREFIX
    AFILE=TLINE(18:23)
    FNAME(1)(1:47)=BFILE
    FNAME(1)(41:47)=AFILE
    TEMP=FNAME(1)
    CALL PACK(TEMP,J)
    FLAG2=.FALSE.
    FLAG1=.FALSE.

C   STORE OFF THE FILE NUMBER XX I.E. ARCHXX
    OUT(CNT)(1:80)=' '
    OUT(CNT)(1:3)=AFILE(5:6)

C   INPUT THE HEADER LINE OF THE ARCHIVE FILE
    OPEN (AUNIT,FILE=TEMP,STATUS='OLD')
    READ(AUNIT,'(A80)') TLINE
    CLOSE (AUNIT)

C   STORE OFF THE HEADER INFORMATION TO DISPLAY ALONG WITH THE FILE NO.
    OUT(CNT)(6:80)=TLINE

C   CAN ONLY SHOW 19 ARCHIVE FILE NAMES AT ONCE ON THE SCREEN
    IF (CNT .EQ. 19) THEN

C       OUTPUT THE 19 FILE NAMES
        CALL MENUWR(RC,20,2,19,OUT,0,2,ST)

C       INPUT THE USERS SELECTION
45      INP(1)=' '
        CALL MENURD(RC,20,1,1,INP,ITR)

C       USER SELECTED TO RETURN, SO CLOSE THE LISTING FILE
        IF (INP(1) .EQ. 'X ') THEN
            CLOSE (BUNIT)
            RETURN
            ENDIF

C       USER SELECTED TO CONTINUE LOOKING AT THE LIST
        IF (INP(1) .EQ. 'C ') THEN
            FLAG1=.TRUE.
            CALL MENUSV(SMFILE,115,RC,20,SMUNIT)
            CNT=2
            GOTO 20
            ENDIF

C       USER SELECTED A NUMBER MAYBE, CHECK TO BE SURE AND ALSO CHECK
C       THAT THE ARCHIVE FILE THE USER PICKED CAN BE OPENED
        READ(INP,'(BN,13)',ERR=35) JJ

C       FORM THE FILE INCLUDING THE DIRECTORY
        AFILE=EFILE(JJ)
        FNAME(1)(1:47)=BFILE
        FNAME(1)(41:47)=AFILE
        TEMP=FNAME(1)
        CALL PACK(TEMP,J)
        INQUIRE (FILE=TEMP,EXIST=FLAG3)
        IF (.NOT. FLAG3) GOTO 45

```

```

GOTO 1
END
SUBROUTINE EVENTA(AFLAG,AUNIT,AFILE,BFILE)

```

```

C*****
C      THIS SUBROUTINE OPENS AN EVENT ARCHIVE FILE AND WRITES AN 80-BYTE
C      HEADER TO IDENTIFY THE FILE.
C
C      HEADER FORMAT:  ARCHIVE FILE: ARCHXX   HH:MM:SS   MM/DD/YY
C
C      VARIABLES PASSED
C
C      AUNIT   -   UNIT FOR OPEN ARCHIVE FILE
C      AFILE   -   ARCHIVE FILE NAME, E.G., ARCH0, ARCH1, ....ARCH99
C      BFILE   -   CROMIX DIRECTORY CONTAINING ARCHIVE FILES AND FILE LIST
C
C      VARAIBLES RETURNED
C
C      AFLAG   -   FALSE:  COULD NOT OPEN AFILE (99 FILES ARLREADY EXIST)
C                  TRUE :  FILE WAS OPEN SUCCESSFULLY
C*****
C
C      CHARACTER*80  ASTAMP(1)
C      CHARACTER*40  BFILE
C      CHARACTER*47  FNAME
C      CHARACTER*15  STAMP
C      CHARACTER*7   AFILE,EFILE
C
C      LOGICAL       AFLAG,FLAG
C
C      INTEGER       AUNIT
C
C      ASTAMP(1)(1:13)= 'ARCHIVE FILE:'
C      ASTAMP(1)(14:80)=' '
C
C      SEE IF AN ARCHIVE FILE CAN BE OPENED.  IF SO OPEN IT AS
C      ARCHX, X=0,1,2,....,99
C      ADD A 54-CHARACTER HEADER IN THE FILE WHICH GIVES THE FILE NAME
C      AND TIME - DATE OF CREATION.
C      DO 10 I=0,99,1
C          AFILE=EFILE(I)
C          FNAME(1:47)=BFILE
C          FNAME(41:47)=AFILE
C          CALL PACK(FNAME,J)
C          INQUIRE (FILE=FNAME,EXIST=FLAG)
C          IF (.NOT. FLAG) THEN
C              AFLAG=.TRUE.
C              OPEN (AUNIT,FILE=FNAME,STATUS='NEW')
C              ASTAMP(1)(14:20)=AFILE
C              CALL DATE(STAMP)
C              ASTAMP(1)(23:37)=STAMP
C              CALL TIME(STAMP)
C              ASTAMP(1)(40:54)=STAMP
C              WRITE(AUNIT,'(A80)') ASTAMP(1)
C              RETURN
C          ENDIF
C
10      CONTINUE
C
AFLAG=.FALSE.

```

```

C   READ IN ALL THE SYSTEM PARAMETERS FROM SYSPARM THAT THE PROGRAM
C   WILL NEED TO EXECUTE.
    OPEN (60, FILE='SYSPARM', STATUS='OLD')
    READ(60, *) ITR, AUNIT, AUNIT1, HUNIT, SUNIT1, SUNIT2, SUNIT3, SUNIT4,
*             CUNIT, CFILE, SFILE1, SFILE2, SFILE3, SFILE4, HFILE, SHFILE,
*             PUNIT1, PUNIT2, PFILE1, PFILE, SMUNIT, SMFILE, BUNIT, BFILE,
*             MUNIT, MFILE
    CLOSE (60)

C   SET UP DEFAULTS
    AFLAG=.FALSE.
    MREAD=.TRUE.

C   DISPLAY THE MAIN MENU
    CALL ONOFF(0)
1   CALL MENUSV(SMFILE, 100, RC, 2, SMUNIT)
2   CMD(1)=' '
    CALL MENURD(RC, 2, 1, 1, CMD, ITR)

C   CHECK FOR A VALID INPUT
    IF (INDEX('123456X', CMD(1)) .EQ. 0) THEN
        CALL MESS(11, RC(2, 1), RC(2, 2), RC(2, 3), 6)
        GOTO 2
    ENDIF

    IF (CMD(1) .EQ. '1') THEN
        IF (AFLAG) THEN
            CALL MESS(12, RC(2, 1), RC(2, 2), RC(2, 3), 6)
        ELSE
            CALL EVENTA(AFLAG, AUNIT, AFILE, BFILE)
            IF (AFLAG) THEN
                CALL MESS(13, RC(2, 1), RC(2, 2), RC(2, 3), 6)
            ELSE
                CALL MESS(14, RC(2, 1), RC(2, 2), RC(2, 3), 6)
            ENDIF
        ENDIF
        GOTO 2
    ENDIF

    IF (CMD(1) .EQ. '2') CALL REVENT(ITR, AUNIT1, SMUNIT, SMFILE,
*                               BUNIT, BFILE)
    IF (CMD(1) .EQ. '3') CALL TCCM(ITR, AUNIT, AFLAG, SUNIT1, SUNIT2,
*                               SUNIT3, SUNIT4, CUNIT, CFILE, SFILE1,
*                               SFILE2, SFILE3, SFILE4, SMUNIT, SMFILE,
*                               MUNIT, MFILE)
    IF (CMD(1) .EQ. '4') CALL DBMM(ITR, CUNIT, SUNIT1, SUNIT2, SUNIT3,
*                               SUNIT4, CFILE, SFILE1, SFILE2, SFILE3,
*                               SFILE4, SHFILE, PUNIT1, PUNIT2, PFILE1,
*                               PFILE, SMUNIT, SMFILE)
    IF (CMD(1) .EQ. '5') THEN
        CALL TIME(STAMP)
        CALL MENUDR(STAMP, RC(2, 1), RC(2, 2), 7, 0, 1, 1)
        GOTO 2
    ENDIF

    IF (CMD(1) .EQ. '6') CALL HELP(ITR, HUNIT, HFILE, SMUNIT, SMFILE)
    IF (CMD(1) .EQ. 'X') THEN
        IF (AFLAG) CLOSE (AUNIT)
        CALL CLEAR(7, 0)
        CALL ONOFF(1)
        STOP
    ENDIF

```

```

      CHARACTER*(*)  STR

      INTEGER        N,I

C     SEARCH THE STRING BACKWARDS FOR A NON BLANK CHARACTER
      DO 5 I=N,2,-1
        IF (STR(I:I) .NE. ' ') GOTO 10
5     CONTINUE

C     RETURN THE NON BLANK POSITION
10    BNDX=I
      RETURN
      END
      SUBROUTINE HELP(ITR,HUNIT,HFILE,SMUNIT,SMFILE)

C*****
C     THIS SUBROUTINE DISPLAYS THE HELP FILE.  THE OPTION OF THE HELP FILE TO
C     BE DISPLAYED IS DETERMINED BY THE USER INPUT.  THE HELP FILES ARE SPLIT
C     UP BY SPECIAL DELIMITERS.  THEY ARE ***&&, WHERE # IDENTIFIES THE PART
C     OF THE FILE THE USER WANTS TO SEE AND && IS THE LENGTH OF THE TEXT THAT
C     PRETAINS TO THAT PORTION OF THE FILE.
C     VARIABLES PASSED:
C
C     ITR      -  INTERACTIVE TERMINAL READ UNIT
C     HUNIT    -  UNIT # TO OPEN THE HELP FILE (HFILE) ON
C     HFILE    -  SYSTEM LEVEL HELP FILE NAME
C     SMUNIT   -  UNIT # TO OPEN THE MENU FILE (SMFILE) ON
C     SMFILE   -  MENU FILE NAME
C*****

      CHARACTER*80  TLINE,LINE
      CHARACTER*7   HFILE,SMFILE
      CHARACTER*1   CMD(1)

      INTEGER       ITR,HUNIT,SMUNIT,RC(3,3)

      LOGICAL       FLAG

C     DISPLAY THE MAIN MENU, AND GET THE USER INPUT
1     CALL MENUSV(SMFILE,125,RC,3,SMUNIT)
2     CMD(1)=' '
      CALL MENURD(RC,3,1,1,CMD,ITR)

C     CHECK FOR A VALID INPUT
      IF (INDEX('12345X',CMD(1)) .EQ. 0) THEN
        CALL MESS(11,RC(2,1),RC(2,2),RC(2,3),6)
        GOTO 2
      ENDIF

C     THE USER SELECTED TO RETURN
      IF (CMD(1) .EQ. 'X') RETURN

C     CHECK TO SEE THAT THE HELP FILE PASSED EXITS, IF NOT DISPLAY AN ERROR
C     MESSAGE THAT HELP IS NOT AVAILABLE AND GO AND GET THE USERS INPUT
      INQUIRE (FILE=HFILE,EXIST=FLAG)
      IF (.NOT. FLAG) THEN
        CALL MESS(15,RC(2,1),RC(2,2),RC(2,3),6)
        GOTO 2
      ENDIF

```


PROGRAM MAPDIG

C THIS PROGRAM ALLOWS THE USER TO EXTERNALLY DIGITIZE MAPS

CHARACTER*1 CMD(1)
CHARACTER*7 SMFILE, MHFILE, MFILE

INTEGER ITR, SMUNIT, RC(2,3), MUNIT, MUNIT1, MUNIT2, MUNIT3,
* ITD

C*****
SMFILE='SYSMENU'
MHFILE='MHELP '
MFILE='MAP.DB '
MUNIT=43
MUNIT1=40
MUNIT2=41
MUNIT3=42
SMUNIT=44
ITR=0
ITD=0
CALL ONOFF(0)

C*****

1 CALL MENUSV(SMFILE, 130, RC, 2, SMUNIT)
2 CMD(1)= ' '
CALL MENURD(RC, 2, 1, 1, CMD, ITR)

IF (INDEX('13X', CMD(1)) .EQ. 0) THEN
CALL MESS(11, RC(2, 1), RC(2, 2), RC(2, 3), 6)
GOTO 2
ENDIF

IF (CMD(1) .EQ. '1') CALL MHELP(ITR, MUNIT, MHFILE, SMUNIT, SMFILE)
IF (CMD(1) .EQ. '3') CALL MADD(ITR, MUNIT1, MUNIT2, MUNIT3, MFILE,
* SMUNIT, SMFILE, ITD)
IF (CMD(1) .EQ. 'X') THEN
CALL CLEAR(7, 0)
CALL ONOFF(1)
STOP
ENDIF

GOTO 1
END

SUBROUTINE MHELP(ITR, MUNIT, MHFILE, SMUNIT, SMFILE)

CHARACTER*80 TLINE, LINE
CHARACTER*7 MHFILE, SMFILE
CHARACTER*1 CMD(1)

INTEGER ITR, MUNIT, SMUNIT, RC(3,3)

LOGICAL FLAG

1 CALL MENUSV(SMFILE, 135, RC, 3, SMUNIT)
2 CMD(1)= ' '
CALL MENURD(RC, 3, 1, 1, CMD, ITR)

C CHECK FOR A VALID INPUT
IF (INDEX('2X', CMD(1)) .EQ. 0) THEN

```

CALL MESS(11, RC(2, 1), RC(2, 2), RC(2, 3), 7)
GOTO 2
ENDIF

IF (CMD(1) .EQ. 'X') RETURN
INQUIRE (FILE=MHFILE, EXIST=FLAG)
IF (.NOT. FLAG) THEN
    CALL MESS(15, RC(2, 1), RC(2, 2), RC(2, 3), 6)
    GOTO 2
ENDIF

CALL CLEAR(7, 0)
OPEN (MUNIT, FILE=MHFILE, STATUS='OLD')
TLINE(1:80)='*X*'
TLINE(2:2)=CMD(1)
5 READ(MUNIT, '(A80)') LINE
IF (TLINE(1:3) .EQ. LINE(1:3)) THEN
    READ(LINE(4:5), '(I2)') IG
    J=0
    DO 20 I=1, IG, 1
        READ(MUNIT, '(A80)') LINE
        J=J+1
        CALL MENUOR(LINE, J, 1, 2, 0, 1, 1)
        IF (MOD(I, 22) .EQ. 0) THEN
            IF (I .EQ. IG) THEN
                CALL MESS(19, RC(3, 1), RC(3, 2), RC(3, 3), 7)
                READ(ITR, '(A1)') CMD(1)
                CLOSE (MUNIT)
                GOTO 1
            ELSE
                CALL MESS(16, RC(3, 1), RC(3, 2), RC(3, 3), 7)
                READ(ITR, '(A1)') CMD(1)
                CALL CLEAR(7, 0)
                J=0
            ENDIF
        ENDIF
    ENDIF
20 CONTINUE
    CALL MESS(19, RC(3, 1), RC(3, 2), RC(3, 3), 7)
    READ(ITR, '(A1)') CMD(1)
    CLOSE (MUNIT)
    GOTO 1
ENDIF

GOTO 5
END
SUBROUTINE MADD(ITR, MUNIT1, MUNIT2, MUNIT3, MFILE, SMUNIT, SMFILE, ITD)

CHARACTER*48 B48
CHARACTER*40 FNAME(3), MNAME, MNAME1, MNAME2, MNAME3, MNAMEX, B40
CHARACTER*7 SMFILE, MFILE
CHARACTER*1 CMD(1), INP

INTEGER RC(5, 3), SMUNIT, MUNIT1, MUNIT2, ITR, ST(3), HDR, MUNIT3,
* HDR1, HDR2, ITD

LOGICAL FLAG, FLAG1, FLAG2

DATA ST/0, 0, 0/

FNAME(1)(1:40)=' '

```

```

FNAME(2)(1:40)=' '
FNAME(3)(1:40)=' '
INP=' '
ITT=-1
5  CALL MENUHV(SMFILE,199,RC,5,SMUNIT)
   ST(3)=0
   CALL MENUWR(RC,5,1,3,FNAME,0,1,ST)

C  FLAG TO GO TO COMMAND LINE AFTER FIRST TIME
   IF (INP .EQ. '&') GOTO 10

   IST=1
15  CALL MENURD(RC,5,IST,3,FNAME,ITR)

10  CMD(1)=' '
   CALL MENURD(RC,5,4,4,CMD,ITT)
   IF (CMD(1) .EQ. ' ') THEN
       ST(3)=1
       CALL MENUWR(RC,5,4,4,CMD,0,1,ST)
       CALL MESS(4,RC(5,1),RC(5,2),RC(5,3),1)
       IST=1
       GOTO 15
   ENDIF
   IF (CMD(1) .EQ. 'X') RETURN
   IF (CMD(1) .NE. 'C') GOTO 10

C  CHECK THAT NONE OF THE FILE NAMES ARE BLANK
   DO 20 K=1,3,1
       IF (FNAME(K) .EQ. ' ') THEN
           ST(3)=1
           CMD(1)=' '
           CALL MENUWR(RC,5,4,4,CMD,0,1,ST)
           CALL MESS(1,RC(5,1),RC(5,2),RC(5,3),7)
           IST=K
           GOTO 15
       ENDIF
20  CONTINUE

C  CHECK THAT THE SYMBOL FILE EXISTS
   MNAME3=FNAME(3)
   CALL PACK(MNAME3,J)
   INQUIRE (FILE=MNAME3,EXIST=FLAG)
   IF (.NOT. FLAG) THEN
       ST(3)=1
       CMD(1)=' '
       CALL MENUWR(RC,5,4,4,CMD,0,1,ST)
       CALL MESS(18,RC(5,1),RC(5,2),RC(5,3),7)
       IST=3
       GOTO 15
   ENDIF

C  SET UP THE MAP.DB DATA BASE IF NECESSARY
   INQUIRE (FILE=MFILE,EXIST=FLAG)
   IF (.NOT. FLAG) THEN
       OPEN (MUNIT3,FILE=MFILE,STATUS='NEW',ACCESS='DIRECT',RECL=42,
*        FORM='FORMATTED')
       HDR=1
       B40(1:40)=' '
       WRITE(B40(1:4),'(I4)') HDR
       WRITE(MUNIT3,'(A40)',REC=1) B40

```

```

        WRITE (MUNIT3, '(A40)', REC=2) B40
        CLOSE (MUNIT3)
    ENDIF

C    SET UP THE BASE MAP DATA BASE IF NECESSARY
    FLAG1=.FALSE.
    MNAME1=FNAME(1)
    CALL PACK(MNAME1,J)
    INQUIRE (FILE=MNAME1, EXIST=FLAG)
    IF (.NOT. FLAG) THEN
        OPEN (MUNIT1, FILE=MNAME1, STATUS='NEW', ACCESS='DIRECT',
*         FORM='FORMATTED', RECL=50)
        HDR=1
        B48(1:48)=' '
        WRITE(B48(1:4), '(I4)') HDR
        WRITE(MUNIT1, '(A48)', REC=1) B48
        WRITE(MUNIT1, '(A48)', REC=2) B48
        CLOSE (MUNIT1)
        FLAG1=.TRUE.
    ENDIF

    MNAME2=MNAME1
    IF ((J+1) .GT. 38) THEN
        MNAME2(38:40)='.MM'
    ELSE
        MNAME2(J+1:J+3)='.MM'
    ENDIF

C    SET UP THE MAP DATA BASE IF NECESSARY
    FLAG2=.FALSE.
    INQUIRE (FILE=MNAME2, EXIST=FLAG)
    IF (.NOT. FLAG) THEN
        OPEN (MUNIT2, FILE=MNAME2, STATUS='NEW', ACCESS='DIRECT', RECL=42,
*         FORM='FORMATTED')
        HDR=1
        B40(1:40)=' '
        WRITE(B40(1:4), '(I4)') HDR
        WRITE(MUNIT2, '(A40)', REC=1) B40
        WRITE(MUNIT2, '(A40)', REC=2) B40
        CLOSE (MUNIT2)
        FLAG2=.TRUE.
    ENDIF

C    ADD THE BASE MAP NAME TO MAP.DB
    OPEN (MUNIT3, FILE=MFILE, STATUS='OLD', ACCESS='DIRECT',
*     FORM='FORMATTED', RECL=42)
    READ(MUNIT3, '(A40)', REC=1) B40
    READ(B40(1:4), '(I4)') HDR
    DO 25 K=2, HDR, 1
        READ(MUNIT3, '(A40)', REC=K) MNAME
        CALL PACK(MNAME, J)
        IF (MNAME .EQ. MNAME1) GOTO 30
25    CONTINUE
    HDR=HDR+1
    WRITE(MUNIT3, '(A40)', REC=HDR) FNAME(1)
    III=HDR+1
    WRITE(MUNIT3, '(A40)', REC=III) FNAME(1)

C    OPEN THE BASE MAP DATA BASE AND THE MAP DATA BASE
30    OPEN (MUNIT1, FILE=MNAME1, STATUS='OLD', ACCESS='DIRECT',
*     FORM='FORMATTED', RECL=50)

```

```

* OPEN (MUNIT2, FILE=MNAME2, STATUS='OLD', ACCESS='DIRECT',
      FORM='FORMATTED', RECL=42)

```

```

MNAMEX=FNAME(2)
CALL PACK(MNAMEX, J)
C CHECK THAT THE MAP FILE NOT ALREADY THERE
READ(MUNIT1, '(A48)', REC=1) B48
READ(B48(1:4), '(I4)') HDR1
DO 35 K=2, HDR1, 1
  READ(MUNIT1, '(A40)', REC=K) MNAME
  CALL PACK(MNAME, J)
  IF (MNAME .EQ. MNAMEX) THEN
    ST(3)=1
    CMD(1)=' '
    CALL MENUWR(RC, 5, 4, 4, CMD, 0, 1, ST)
    CALL MESS(2, RC(5, 1), RC(5, 2), RC(5, 3), 7)
    IST=2
    GOTO 15

```

```

ENDIF

```

```

35 CONTINUE
HDR1=HDR1+1
READ(MUNIT2, '(A40)', REC=1) B40
READ(B40(1:4), '(I4)') HDR2
HDR2=HDR2+1
IS=HDR2

```

```

CALL MDIG(ITR, SMUNIT, SMFILE, MUNIT2, HDR2, IERR, ITD, FNAME(?),
* FNAME(3))

```

```

IF (IERR .EQ. 0) THEN
  IE=HDR2
  B48(1:48)=' '
  WRITE(B48(1:4), '(I4)') HDR1
  WRITE(MUNIT1, '(A48)', REC=1) B48
  B48(1:48)=' '
  B48(1:40)=FNAME(2)
  WRITE(B48(41:48), '(2I4)') IS, IE
  WRITE(MUNIT1, '(A48)', REC=HDR1) B48
  III=HDR1+1
  WRITE(MUNIT1, '(A48)', REC=III) B48
  B40(1:40)=' '
  WRITE(B40(1:4), '(I4)') HDR2
  WRITE(MUNIT2, '(A40)', REC=1) B40
  WRITE(MUNIT2, '(A40)', REC=IE+1) B40
  B40(1:40)=' '
  WRITE(B40(1:4), '(I4)') HDR
  WRITE(MUNIT3, '(A40)', REC=1) B40
ENDIF

```

```

IF ((IERR .NE. 0).AND. (FLAG1)) THEN
  CLOSE (MUNIT1, STATUS='DELETE')
ELSE
  CLOSE (MUNIT1)
ENDIF
IF ((IERR .NE. 0).AND. (FLAG2)) THEN
  CLOSE (MUNIT2, STATUS='DELETE')
ELSE
  CLOSE (MUNIT2)
ENDIF

```

```

CLOSE (MUNIT3)
INP=' '

```

```

GOTO 5

END
SUBROUTINE PACK(STR,J)

CHARACTER*(*) STR

INTEGER      I,J

L=LEN(STR)
J=0

DO 5 I=1,L,1
  IF (STR(I:I) .NE. ' ') THEN
    J=J+1
    STR(J:J)=STR(I:I)
  ENDIF
5  CONTINUE

DO 10 I=J+1,L,1
  STR(I:I)=' '
10 CONTINUE

RETURN
END
SUBROUTINE MDIG(ITR,SMUNIT,SMFILE,MUNIT2,HDR2,IERR,ITD,
*             MFILE,SFILE)

CHARACTER*40 TEXT(4),SFILE,MFILE,TSTAMP,TFILE
CHARACTER*30 MES(1)
CHARACTER*20 IOP,ANNO(1)
CHARACTER*15 TEMP
CHARACTER*10 DIG(2)
CHARACTER*7  SMFILE
CHARACTER*6  USRUNT(1)
CHARACTER*2  INP(1)
CHARACTER*1  CMD(1)

INTEGER      ITR,SMUNIT,MUNIT2,HDR2,IERR,ITD,RC(10,3),ST(3),
*             ANGLE(40),WCS(40,3),LINK(8)

REAL         VLX,VLX,VLY,VUX,VUY,ULX,ULY,UUX,UUY,SCALE(40)

DATA ST/0,0,0/

ITT=-1
USRUNT(1)='FEET '

C  INITIALIZE THE MENU VARIABLES
100 TEXT(1)(1:40)=' '
   TEXT(2)(1:40)=' '
   TEXT(3)(1:40)=' '
   TEXT(4)(1:40)=' '

C  DISPLAY THE DIGITIZING MENU
CALL MENUSV(SMFILE,200,RC,10,SMUNIT)

C  DISPLAY THE USER UNITS
ST(3)=1
CALL MENUWR(RC,10,3,3,USRUNT,0,1,ST)

```

```

C      READ IN THE TITLE
      CALL MENURD(RC,10,1,2,TEXT,ITR)

C      INPUT THE USER UNITS
      DIG(1)(1:10)=' '
      DIG(2)(1:10)=' '
      IST=4
301    CALL MENURD(RC,10,IST,5,DIG,ITT)
      READ(DIG(1)(1:10),'(F10.0)',ERR=302) ULX
      READ(DIG(2)(1:10),'(F10.0)',ERR=303) ULY
      GOTO 400
302    IST=4
      GOTO 301
303    IST=5
      GOTO 301

C      INPUT THE VIRTUAL UNITS FROM THE DIGITIZER
400    MES(1)='DIGITIZE LOWER LEFT POINT'
      CALL MENUWR(RC,10,8,8,MES,0,7,ST)
      READ(ITD,*) IBUT,VLX,VLX
      MES(1)(1:30)=' '
      CALL MENUWR(RC,10,8,8,MES,0,1,ST)

C      INPUT THE USER UNITS
      DIG(1)(1:10)=' '
      DIG(2)(1:10)=' '
      IST=6
310    CALL MENURD(RC,10,IST,7,DIG,ITT)
      READ(DIG(1)(1:10),'(F10.0)',ERR=311) UUX
      READ(DIG(2)(1:10),'(F10.0)',ERR=312) UUY
      GOTO 500
311    IST=6
      GOTO 310
312    IST=7
      GOTO 310

C      INPUT THE VIRTUAL UNITS FROM THE DIGITIZER
500    MES(1)='DIGITIZE UPPER RIGHT POINT'
      CALL MENUWR(RC,10,9,9,MES,0,7,ST)
      READ(ITD,*) IBUT,VUX,VUY
      MES(1)(1:30)=' '
      CALL MENUWR(RC,10,9,9,MES,0,1,ST)

C      SCAN THE COMMAND INPUT LINE
200    CMD(1)=' '
      CALL MENURD(RC,10,10,10,CMD,ITT)
      IF (CMD(1).EQ. 'X') THEN
          IERR=1
          RETURN
      ENDIF
      IF (CMD(1).EQ. 'R') GOTO 100
      IF (CMD(1).NE. 'C') GOTO 200

C      FILL IN THE HEADER INFORMATION OF THE FILE
      WRITE(MUNIT2,'(A40)',REC=HDR2) MFILE

C      WRITE THE MAP DESCRIPTION IN THE HEADER
      WRITE(MUNIT2,'(A40)',REC=HDR2+1) TEXT(1)
      WRITE(MUNIT2,'(A40)',REC=HDR2+2) TEXT(2)

```

```

WRITE(MUNIT2,'(A40)',REC=HDR2+3) TEXT(3)
WRITE(MUNIT2,'(A40)',REC=HDR2+4) TEXT(4)

C   WRITE THE SYMBOL FILE NAME IN THE HEADER
WRITE(MUNIT2,'(A40)',REC=HDR2+5) SFILE

C   OBTAIN THE TIME STAMP FOR CREATION DATE
TSTAMP(1:40)=' '
CALL TIME(TEMP)
TSTAMP(1:10)=TEMP(1:10)
CALL DATE(TEMP)
TSTAMP(11:20)=TEMP(1:10)

C   WRITE THE TIME AND DATE STAMP IN THE HEADER
WRITE(MUNIT2,'(A40)',REC=HDR2+6) TSTAMP

C   STORE THE VIRTUAL AND USER UNITS IN THE HEADER
WRITE(TEXT(4)(1:40),'(4F10.2)') VLX,VLY,ULX,ULY
WRITE(MUNIT2,'(A40)',REC=HDR2+7) TEXT(4)

C   STORE THE VIRTUAL AND USER UNITS IN THE HEADER
WRITE(TEXT(4)(1:40),'(4F10.2)') VUX,VUY,UUX,UUY
WRITE(MUNIT2,'(A40)',REC=HDR2+8) TEXT(4)

C   OUTPUT THE USER UNITS, SUB UNITS ...
TEXT(4)(1:40)=' '
TEXT(4)(1:6)=USRUNT(1)
TEXT(4)(7:12)=' INCHES'
TEXT(4)(13:16)=' 12'
TEXT(4)(17:20)=' 0000'
WRITE(MUNIT2,'(A40)',REC=HDR2+9) TEXT(4)

C   BLANK OUT THE UNUSED HEADER RECORDS
TEXT(4)(1:40)=' '
WRITE(MUNIT2,'(A40)',REC=HDR2+10) TEXT(4)
WRITE(MUNIT2,'(A40)',REC=HDR2+11) TEXT(4)
WRITE(MUNIT2,'(A40)',REC=HDR2+12) TEXT(4)
WRITE(MUNIT2,'(A40)',REC=HDR2+13) TEXT(4)
WRITE(MUNIT2,'(A40)',REC=HDR2+14) TEXT(4)

C   POINT TO THE LAST RECORD
HDR2=HDR2+14

C   SET UP THE DEFAULT WEIGHT, COLOR, STYLE, SCALE AND ANGLE
DO 599 I=1,40,1
    WCS(I,1)=0
    WCS(I,2)=1
    WCS(I,3)=0
    SCALE(I)=1.
    ANGLE(I)=0
599  CONTINUE

TFILE=SFILE
CALL PACK(TFILE,J)

C   READ IN THE SYMBOL FILE AND SET THE WEIGHT, STYLE, ...
OPEN (SMUNIT,FILE=TFILE,STATUS='OLD',ACCESS='DIRECT',RECL=20,
*    FORM='FORMATTED')
READ(SMUNIT,'(I2)',REC=1) IHD
DO 598 I=1,IHD,1
    READ(SMUNIT,'(A20)',REC=I+1) IOP
    READ(IOP(1:2),'(I2)') II

```

```

      READ(IOP(3:16), '(3I2,F4.2,I4)') WCS(II,1),WCS(II,2),WCS(II,3),
*      SCALE(II),ANGLE(II)
598  CONTINUE
      CLOSE (SMUNIT)

C    DISPLAY THE ENTRY MENU
600  LINK(1)=0
      LINK(2)=0
      LINK(3)=0
      LINK(4)=0
      LINK(5)=0
      LINK(6)=0
      LINK(7)=0
      LINK(8)=0
      CALL MENUSV(SMFILE,210,RC,10,SMUNIT)

C    INPUT THE ANNOTATION
      ANNO(1)(1:20)=' '
      CALL MENURD(RC,10,1,1,ANNO,ITR)
      AX=0.
      AY=0.
      IF (ANNO(1) .NE. ' ') THEN
        TEXT(1)='DIGITIZE ANNOTATION PLACEMENT'
        ST(3)=1
        CALL MENUWR(RC,10,2,2,TEXT,0,7,ST)
        READ(ITD,*) IBUT,AX,AY
        ENDIF

C    SCAN THE COMMAND INPUT LINE
300  INP(1)=' '
      CALL MENURD(RC,10,3,3,INP,ITT)
      IF (INP(1) .EQ. 'X ') THEN
        IERR=0
        RETURN
        ENDIF

C    CHECK FOR A VALID INPUT
      READ(INP(1)(1:2), '(I2)',ERR=300) II
      IF ((II .LE. 0).OR.(II .GE. 21)) GOTO 300

C    IF POINT FIND OUT IF POTENTIAL SOURCE
      IF ((II .EQ. 1).OR.(II .EQ. 2).OR.(II .EQ. 3).OR.(II .EQ. 8).OR.
*      (II .EQ. 16).OR.(II .EQ. 17)) THEN
        CMD(1)=' '
        CALL MENURD(RC,10,4,4,CMD,ITT)
        LINK(1)=0
        IF (CMD(1) .EQ. 'Y') LINK(1)=1
        ENDIF

C    ENTER THE MAP DATA
      CALL MAPDAT(ITR,SMUNIT,SMFILE,MUNIT2,HDR2,ITD,II,WCS,SCALE,
*      ANGLE,40,ANNO(1),AX,AY,LINK)

      GOTO 600
      END
      SUBROUTINE MAPDAT(ITR,SMUNIT,SMFILE,MUNIT,HDR,ITD,II,WCS,SCALE,
*      ANGLE,N,ANNO,AX,AY,LINK)

      CHARACTER*40 IO
      CHARACTER*20 ANNO
      CHARACTER*7  SMFILE

```

```

      INTEGER      ITR, SMUNIT, MUNIT, HDR, ITD, II, ANGLE(N), WCS(N, 3),
*                LINK(8)

      REAL          VX, VY, VX1, VY1, SCALE(N), VXMIN, VYMIN, VXMAX, VYMAX,
*                AX, AY

      LOGICAL       FLAG

```

FLAG=.TRUE.

C ENTRY WILL BE A POINT

```

      IF ((II .EQ. 1).OR.(II .EQ. 2).OR.(II .EQ. 3).OR.(II .EQ. 8).OR.
*       (II .EQ. 16).OR.(II .EQ. 17)) THEN
        IO(1:40)=' '
        IO(1:2)='10'
        WRITE(IO(3:4), '(I2)') II
        IO(5:6)=' 1'
        WRITE(IO(7:20), '(3I2,F4.2,I4)') WCS(II,1),WCS(II,2),WCS(II,3),
*                                     SCALE(II),ANGLE(II)

        IO(21:24)='0000'
        WRITE(IO(25:40), '(8I2)') LINK
        WRITE(MUNIT, '(A40)', REC=HDR+1) IO
        WRITE(IO(1:20), '(2F10.2)') AX, AY
        IO(21:40)=ANNO
        WRITE(MUNIT, '(A40)', REC=HDR+2) IO
        READ(ITD,*) IBUT, VX, VY
        IO(1:40)=' '
        WRITE(IO(1:20), '(2F10.2)') VX, VY
        WRITE(MUNIT, '(A40)', REC=HDR+3) IO
        HDR=HDR+3

```

ENDIF

C ENTRY WILL BE A LINE

```

      IF ((II .EQ. 11).OR.(II .EQ. 12).OR.(II .EQ. 13).OR.
*       (II .EQ. 14).OR.(II .EQ. 18).OR.(II .EQ. 19).OR.
*       (II .EQ. 20)) THEN
        IO(1:40)=' '
        IO(1:2)='11'
        WRITE(IO(3:4), '(I2)') II
        WRITE(IO(7:20), '(3I2,F4.2,I4)') WCS(II,1),WCS(II,2),WCS(II,3),
*                                     SCALE(II),ANGLE(II)

        IO(21:24)='0000'
        WRITE(IO(25:40), '(8I2)') LINK
        WRITE(MUNIT, '(A40)', REC=HDR+1) IO
        WRITE(IO(1:20), '(2F10.2)') AX, AY
        IO(21:40)=ANNO
        WRITE(MUNIT, '(A40)', REC=HDR+2) IO
        IO(1:40)=' '
        WRITE(MUNIT, '(A40)', REC=HDR+3) IO
        III=HDR+3
        NPTS=0
        READ(ITD,*) IBUT, VX, VY
        IF (IBUT .EQ. 9) THEN
          READ(MUNIT, '(A40)', REC=HDR+1) IO
          WRITE(IO(5:6), '(I2)') NPTS
          WRITE(MUNIT, '(A40)', REC=HDR+1) IO
          WRITE(IO(1:40), '(4F10.2)') VXMIN, VXMAX, VYMIN, VYMAX
          WRITE(MUNIT, '(A40)', REC=HDR+3) IO
          HDR=HDR+3+(NPTS/2)
          RETURN

```

100

```

                                ENDIF
IF (FLAG) THEN
    FLAG=.FALSE.
    VXMIN=VX
    VXMAX=VX
    VYMIN=VY
    VYMAX=VY
ENDIF
IF (VX .GT. VXMAX) VXMAX=VX
IF (VX .LT. VXMIN) VXMIN=VX
IF (VY .GT. VYMAX) VYMAX=VY
IF (VY .LT. VYMIN) VYMIN=VY
READ(ITD,*) IBUT,VX1,VY1
IF (IBUT .EQ. 9) THEN
    IO(1:40)=' '
    WRITE(IO(1:20),'(2F10.2)') VX,VY
    JJJ=III+(NPTS/2)+1
    WRITE(MUNIT,'(A40)',REC=JJJ) IO
    READ(MUNIT,'(A40)',REC=HDR+1) IO
    NPTS=NPTS+1
    WRITE(IO(5:6),'(I2)') NPTS
    WRITE(MUNIT,'(A40)',REC=HDR+1) IO
    WRITE(IO(1:40),'(4F10.2)') VXMIN,VXMAX,VYMIN,VYMAX
    WRITE(MUNIT,'(A40)',REC=HDR+3) IO
    HDR=HDR+3+(NPTS/2)+1
    RETURN
                                ENDIF
IF (VX1 .GT. VXMAX) VXMAX=VX1
IF (VX1 .LT. VXMIN) VXMIN=VX1
IF (VY1 .GT. VYMAX) VYMAX=VY1
IF (VY1 .LT. VYMIN) VYMIN=VY1
WRITE(IO(1:40),'(4F10.2)') VX,VY,VX1,VY1
NPTS=NPTS+2
JJJ=III+(NPTS/2)
WRITE(MUNIT,'(A40)',REC=JJJ) IO
GOTO 100
                                ENDIF

```

```

C      ENTRY WILL BE A POLYGON
      IF ((II .EQ. 4).OR.(II .EQ. 5).OR.(II .EQ. 6).OR.(II .EQ. 7).OR.
*      (II .EQ. 9).OR.(II .EQ. 10).OR.(II .EQ. 15)) THEN
          IO(1:40)=' '
          IO(1:2)='12'
          WRITE(IO(3:4),'(I2)') II
          WRITE(IO(7:20),'(3I2,F4.2,I4)') WCS(II,1),WCS(II,2),WCS(II,3),
*                                     SCALE(II),ANGLE(II)
          IO(21:24)='0000'
          WRITE(IO(25:40),'(8I2)') LINK
          WRITE(MUNIT,'(A40)',REC=HDR+1) IO
          WRITE(IO(1:20),'(2F10.2)') AX,AY
          IO(21:40)=ANNO
          WRITE(MUNIT,'(A40)',REC=HDR+2) IO
          IO(1:40)=' '
          WRITE(MUNIT,'(A40)',REC=HDR+3) IO
          III=HDR+3
          NPTS=0
          READ(ITD,*) IBUT,VX,VY
          IF (IBUT .EQ. 9) THEN
              READ(MUNIT,'(A40)',REC=HDR+1) IO
              WRITE(IO(5:6),'(I2)') NPTS

```

200

```

WRITE(MUNIT,'(A40)',REC=HDR+1) IO
WRITE(IO(1:40),'(4F10.2)') VXMIN,VXMAX,VYMIN,VYMAX
WRITE(MUNIT,'(A40)',REC=HDR+3) IO
HDR=HDR+3+(NPTS/2)
RETURN

```

ENDIF

```

IF (FLAG) THEN

```

```

    FLAG=.FALSE.

```

```

    VXMIN=VX

```

```

    VXMAX=VX

```

```

    VYMIN=VY

```

```

    VYMAX=VY

```

```

ENDIF

```

```

IF (VX .GT. VXMAX) VXMAX=VX

```

```

IF (VX .LT. VXMIN) VXMIN=VX

```

```

IF (VY .GT. VYMAX) VYMAX=VY

```

```

IF (VY .LT. VYMIN) VYMIN=VY

```

```

READ(ITD,*) IBUT,VX1,VY1

```

```

IF (IBUT .EQ. 9) THEN

```

```

    IO(1:40)=' '

```

```

    WRITE(IO(1:20),'(2F10.2)') VX,VY

```

```

    JJJ=III+(NPTS/2)+1

```

```

    WRITE(MUNIT,'(A40)',REC=JJJ) IO

```

```

    READ(MUNIT,'(A40)',REC=HDR+1) IO

```

```

    NPTS=NPTS+1

```

```

    WRITE(IO(5:6),'(I2)') NPTS

```

```

    WRITE(MUNIT,'(A40)',REC=HDR+1) IO

```

```

    WRITE(IO(1:40),'(4F10.2)') VXMIN,VXMAX,VYMIN,VYMAX

```

```

    WRITE(MUNIT,'(A40)',REC=HDR+3) IO

```

```

    HDR=HDR+3+(NPTS/2)+1

```

```

    RETURN

```

ENDIF

```

IF (VX1 .GT. VXMAX) VXMAX=VX1

```

```

IF (VX1 .LT. VXMIN) VXMIN=VX1

```

```

IF (VY1 .GT. VYMAX) VYMAX=VY1

```

```

IF (VY1 .LT. VYMIN) VYMIN=VY1

```

```

WRITE(IO(1:40),'(4F10.2)') VX,VY,VX1,VY1

```

```

NPTS=NPTS+2

```

```

JJJ=III+(NPTS/2)

```

```

WRITE(MUNIT,'(A40)',REC=JJJ) IO

```

```

GOTO 200

```

ENDIF

```

RETURN

```

```

END

```

```

SUBROUTINE TIME(STAMP)

```

```

C THIS SUBROUTINE DISPLAYS THE TIME ON THE SCREEN WITH
C THE COMMAND MENU PRESENT

```

```

CHARACTER*15 STAMP

```

```

INTEGER HOUR,MINUTE,SECOND

```

```

COMMON/TIM1/HOUR,MINUTE,SECOND

```

```

C GET THE TIME FROM THE SYSTEM
C CALL RTIME

```

```

STAMP(1:15)=' '

```

```

C GENERATE THE TIME STAMP

```

```

STAMP(3:3) = ':'

```

```

STAMP(6:6) = ':'

```

```

WRITE(Stamp(1:2),1) HOUR
WRITE(Stamp(4:5),1) MINUTE
WRITE(Stamp(7:8),1) SECOND
1  FORMAT(I2)

C  FILL IN ANY BLANKS WITH A 0
IF (Stamp(1:1) .EQ. ' ') Stamp(1:1) = '0'
IF (Stamp(2:2) .EQ. ' ') Stamp(2:2) = '0'
IF (Stamp(4:4) .EQ. ' ') Stamp(4:4) = '0'
IF (Stamp(5:5) .EQ. ' ') Stamp(5:5) = '0'
IF (Stamp(7:7) .EQ. ' ') Stamp(7:7) = '0'
IF (Stamp(8:8) .EQ. ' ') Stamp(8:8) = '0'

RETURN
END
SUBROUTINE DATE(Stamp)

CHARACTER*15  Stamp
INTEGER      MONTH, DAY, YEAR, DOW
COMMON /DAT1/ DOW, YEAR, MONTH, DAY

CALL RDATE

Stamp(1:15)=' '
Stamp(3:3)='/'
Stamp(6:6)='/'
WRITE(Stamp(1:2),5) MONTH
WRITE(Stamp(4:5),5) DAY
WRITE(Stamp(7:8),5) YEAR
5  FORMAT(A2)

RETURN
END

```

```

C      CLEAR THE SCREEN AFTER OBTAINING THE USERS INPUT AND OPEN THE HELP FILE
      CALL CLEAR(7,0)
      OPEN (HUNIT,FILE=HFILE,STATUS='OLD')

C      SET UP THE SEARCH KEY *** TO FIND THE REQUIRED PART OF THE FILE
      TLINE(1:80)='*X*'
      TLINE(2:2)=CMD(1)

5      READ(HUNIT,'(A80)') LINE

C      A MATCH WAS FOUND TO ***
      IF (TLINE(1:3) .EQ. LINE(1:3)) THEN

C          FIND THE NUMBER OF LINE OF TEXT THAT NEEDS TO BE DISPLAYED
          READ(LINE(4:5),'(I2)') IG

          J=0
          DO 20 I=1,IG,1
              READ(HUNIT,'(A80)') LINE
              J=J+1

C              DISPLAY THE LINE OF TEXT ON THE SCREEN
              CALL MENUADR(LINE,J,1,2,0,1,1)

C          CAN ONLY DISPLAY 22 LINES ON THE SCREEN AT A TIME
          IF (J .EQ. 22) THEN

C              DETERMINE IF THERE IS MORE TEXT TO BE PRINTED
              IF (I .EQ. IG) THEN

C                  DISPLAY THE END OF FILE MESSAGE TO INDICATE THAT
C                  THIS IS THE END OF THE HELP FILE
                  CALL MESS(19,RC(3,1),RC(3,2),RC(3,3),6)
                  READ(ITR,'(A1)') CMD(1)
                  CLOSE (HUNIT)
                  GOTO 1

                  ELSE

C                  DISPLAY MESSAGE PRESS RETRUN TO CONTINUE
                  CALL MESS(16,RC(3,1),RC(3,2),RC(3,3),6)
                  READ(ITR,'(A1)') CMD(1)
                  CALL CLEAR(7,0)
                  J=0

                  ENDIF
              ENDIF

20      CONTINUE

C      DISPLAY END OF FILE MESSAGE TO INDICATE NO MORE HELP TEXT
C      AVAILABLE.
      CALL MESS(19,RC(3,1),RC(3,2),RC(3,3),6)
      READ(ITR,'(A1)') CMD(1)
      CLOSE (HUNIT)
      GOTO 1

      ENDIF

C      KEY DID NOT MATCH, GO AND TRY ANOTHER LINE
      GOTO 5
      END
      SUBROUTINE TIME(Stamp)

```

C*****

C THIS SUBROUTINE GETS THE TIME FROM THE SYSTEM AND PASSES IT BACK IN STAMP
 C THE TIME IS PASSED BACK IN THE FORM HH:MM:SS
 C*****

CHARACTER*15 STAMP
 INTEGER HOUR, MINUTE, SECOND, DOW, YEAR, MON, DAY
 COMMON/TIM1/HOUR, MINUTE, SECOND

C GET THE TIME FROM THE SYSTEM, RTIME IS A 68000 ASSEMBLER PROGRAM
 C DATA IS PASSED IN THE COMMON BLOCK TIM1
 CALL RTIME

C GENERATE THE TIME STAMP
 STAMP(1:15)=' '
 STAMP(3:3) = ':'
 STAMP(6:6) = ':'
 WRITE(STAMP(1:2),1) HOUR
 WRITE(STAMP(4:5),1) MINUTE
 WRITE(STAMP(7:8),1) SECOND
 1 FORMAT(I2)

C FILL IN ANY BLANKS WITH A 0
 IF (STAMP(1:1) .EQ. ' ') STAMP(1:1) = '0'
 IF (STAMP(2:2) .EQ. ' ') STAMP(2:2) = '0'
 IF (STAMP(4:4) .EQ. ' ') STAMP(4:4) = '0'
 IF (STAMP(5:5) .EQ. ' ') STAMP(5:5) = '0'
 IF (STAMP(7:7) .EQ. ' ') STAMP(7:7) = '0'
 IF (STAMP(8:8) .EQ. ' ') STAMP(8:8) = '0'

RETURN
 END
 SUBROUTINE DATE(STAMP)

C*****
 C THIS ROUTINE OBTAINS THE DATE FROM THE SYSTEM AND RETURNS IT IN STAMP
 C THE DATE IS RETURNED IN THE FORM MM/DD/YY
 C*****

CHARACTER*15 STAMP
 INTEGER HOUR, MINUTE, SEC, DOW, YEAR, MON, DAY
 COMMON/DAT1/DOW, YEAR, MON, DAY

C GET THE DATE FROM THE SYSTEM, RDATE IS A 68000 ASSEMBLER PROGRAM
 C THE DATA IS PASSED THROUGH THE COMMON BLOCK DAT1
 CALL RDATE

C FORM THE DATE
 STAMP(1:15)=' '
 STAMP(3:3)='/'
 STAMP(6:6)='/'
 WRITE(STAMP(1:2),5) MON
 WRITE(STAMP(4:5),5) DAY
 WRITE(STAMP(7:8),5) YEAR
 5 FORMAT(I2)

RETURN
 END
 SUBROUTINE DBMM(ITR, CUNIT, SUNIT1, SUNIT2, SUNIT3, SUNIT4,
 * CFIL, SFIL1, SFIL2, SFIL3, SFIL4,
 * SHFIL, PUNIT1, PUNIT2, PFIL1, PHFIL,

* SMUNIT, SMFILE)

```
C*****
C      THIS SUBROUTINE MANAGES THE SUBSTANCE-SOURCE AND PROCEDURE DATA BASES
C      VARIABLES PASSED:
C
C      ITR      -  INTERACTIVE TERMINAL READ UNIT
C      SMUNIT   -  UNIT # TO OPEN THE MENU FILE (SMFILE) ON
C      SMFILE   -  MENU FILE NAME
C      SHFILE   -  SUBSTANCE-SOURCE HELP FILE NAME
C      SUNIT1   -  UNIT # TO OPEN SFILE1 ON
C      SUNIT2   -  UNIT # TO OPEN SFILE2 ON
C      SUNIT3   -  UNIT # TO OPEN SFILE3 ON
C      SUNIT4   -  UNIT # TO OPEN SFILE4 ON
C      PHFILE   -  PROCEDURE HELP FILE NAME
C      PUNIT1   -  UNIT # TO OPEN PFILE1
C      PUNIT2   -  UNIT # TO OPEN PROCEDURE FILE OF INTEREST ON
C      PFILE1   -  FILE NAME CONTAINING LIST OF PROCEDURE CURRENT PROCEDURE
C                  FILES
C*****
```

```
      INTEGER      ITR, CUNIT, SUNIT1, SUNIT2, SUNIT3, SUNIT4, PUNIT1, PUNIT2,
*                  SMUNIT, RC(2,3)
```

```
      CHARACTER*1  CMD(1)
      CHARACTER*7  CFILE, SFILE1, SFILE2, SFILE3, SFILE4, SHFILE, PFILE1,
*                  PHFILE, SMFILE
```

```
C      DISPLAY THE MAIN MENU AND INPUT USER SELECTION
5      CALL MENUSV(SMFILE, 120, RC, 2, SMUNIT)
```

```
10     CMD(1)=' '
      CALL MENURD(RC, 2, 1, 1, CMD, ITR)
```

```
C      CHECK FOR VALID INPUT
      IF (INDEX('12X', CMD(1)) .EQ. 0) THEN
          CALL MESS(11, RC(2, 1), RC(2, 2), RC(2, 3), 6)
          GOTO 10
      ENDIF
```

```
C      THE USER SELECTED THE SUBSTANCE-SOURCE DATA BASE
      IF (CMD(1) .EQ. '1') CALL SSSIP(ITR, SUNIT1, SUNIT2, SUNIT3, SUNIT4,
*                                  SFILE1, SFILE2, SFILE3, SFILE4,
*                                  SHFILE, SMUNIT, SMFILE)
```

```
C      THE USER SELECTED THE PROCEDURE DATA BASE
      IF (CMD(1) .EQ. '2') CALL PROCD(ITR, PUNIT1, PUNIT2, PFILE1, PHFILE,
*                                  SMUNIT, SMFILE)
```

```
C      THE USER SELECTED TO RETURN
      IF (CMD(1) .EQ. 'X') RETURN

      GOTO 5
END
```


PROGRAM SYMBOL

CHARACTER*40 FNAME
CHARACTER*20 IO

INTEGER ITW, ITR, SUNIT, SN, WT, COLOR, STYLE, ANGLE

REAL SCALE

ITW=1
ITR=1
SUNIT=14

C INPUT THE SYMBOL FILE NAME
WRITE(ITW, '(/, ''ENTER THE SYMBOL FILE NAME'')')
READ(ITR, '(A40)') FNAME

OPEN (SUNIT, FILE=FNAME, STATUS='NEW', ACCESS='DIRECT', RECL=20,
* FORM='FORMATTED')

IO(1:20)=' '
WRITE(SUNIT, '(A20)', REC=1) IO
ICNT=0
100 WRITE(ITW, '(''ENTER SYMBOL NUMBER , -999 TO QUIT'')')
READ(ITR, *) SN
IF (SN .EQ. -999) THEN
 IO(1:20)=' '
 WRITE(IO(1:2), '(I2)') ICNT
 WRITE(SUNIT, '(A20)', REC=1) IO
 CLOSE (SUNIT)
 STOP
 ENDIF
WRITE(ITW, '(''ENTER WEIGHT'')')
READ(ITR, *) WT
WRITE(ITW, '(''ENTER COLOR'')')
READ(ITR, *) COLOR
WRITE(ITW, '(''ENTER STYLE'')')
READ(ITR, *) STYLE
WRITE(ITW, '(''ENTER SCALE'')')
READ(ITR, '(F10.0)') SCALE
WRITE(ITW, '(''ENTER ANGLE'')')
READ(ITR, *) ANGLE

ICNT=ICNT+1
IO(1:20)=' '
WRITE(IO(1:16), '(4I2, F4.2, I4)') SN, WT, COLOR, STYLE, SCALE, ANGLE
WRITE(SUNIT, '(A20)', REC=ICNT+1) IO
GOTO 100

END

Filename - convert. for

A 10x10 grid of dots. The dots are arranged to form the letters 'L' and 'U'. The 'L' is formed by dots in the first column (all 10 dots) and the first row (all 10 dots). The 'U' is formed by dots in the second, third, and fourth columns (all 10 dots each) and the second, third, and fourth rows (all 10 dots each). The intersection of the 'L' and 'U' is the first row, second, third, and fourth columns.

● ●

A 10x10 grid of dots where the number 10 is formed by removing dots. The number 1 is formed by a vertical column of 10 dots in the 4th column from the left. The number 0 is formed by a rectangular frame of dots, with the top and bottom rows being full 10-dot rows, and the left and right sides being vertical columns of 8 dots each (rows 2 through 9). All other dots in the grid are present.

PROGRAM CNVRT

CHARACTER*80 LINE
 CHARACTER*40 FIN,FOUT
 CHARACTER*20 FTYPE
 CHARACTER*5 FMT
 CHARACTER*1 TYPE

WRITE(0,'(//, ''ENTER FILE TO BE CONVERTED TO DIRECT'')')
 READ(0,'(A40)') FIN

WRITE(0,'(//, ''ENTER NEW FILE NAME OF DIRECT FILE'')')
 READ(0,'(A40)') FOUT

WRITE(0,'(//, ''ENTER DESIRED OUTPUT RECL LENGTH'')')
 READ(0,*) IRECL

WRITE(0,'(//, ''OUTPUT FILE FORMATTED OR UNFORMATTED (F OR U)'')')
 READ(0,'(A1)') TYPE

FTYPE='UNFORMATTED'
 IF (TYPE .EQ. 'F') FTYPE='FORMATTED'
 IREC1=IRECL
 IF (TYPE .EQ. 'F') IREC1=IRECL+2
 OPEN (12,FILE=FIN,STATUS='OLD')
 OPEN (13,FILE=FOUT,STATUS='NEW',ACCESS='DIRECT',RECL=IREC1,
 * FORM=FTYPE)

FMT='(AXX)'
 WRITE(FMT(3:4),'(I2)') IRECL

I=1
 50 READ(12,'(A80)',END=100) LINE
 IF (TYPE .EQ. 'F') THEN
 WRITE(13,FMT,REC=I) LINE(1:IRECL)
 ELSE
 WRITE(13,REC=I) LINE(1:IRECL)
 ENDIF
 I=I+1
 GOTO 50
 100 WRITE(0,'(//,13, '' RECORDS WERE PROCESSED'')') I-1
 CLOSE (12)
 CLOSE (13)
 STOP
 END


```

      INTEGER FUNCTION BNDX(STR,N)

      CHARACTER*(*) STR
      INTEGER      N,I

      DO 5 I=N,2,-1
        IF (STR(I:I) .NE. ' ') GOTO 10
5      CONTINUE

10     BNDX=I
      RETURN
      END

```



```
$segment %menuseg
SUBROUTINE MENUSV(FNAME,MN,RC,N,INU)
```

```
C*****
C THIS SUBROUTINE DISPLAY THE MENUS ON THE SCREEN AND LOADS AN ARRAY WITH THE
C INPUT FIELD DATA. IT ALSO INITIALLY CONVERTS THE FILE TO DIRECT ACCESS THE
C FIRST TIME IT IS CALLED.
```

```
C VARIABLES PASSED:
```

```
C
C FNAME - MENU FILE NAME
C MN - MENU NUMBER TO BE DISPLAYED
C RC - ARRAY TO LOAD WITH THE INPUT FIELD DATA
C RC(N,1) - ROW POSITION OF THE INPUT FIELD
C RC(N,2) - COLUMN POSITION OF THE INPUT FIELD
C RC(N,3) - LENGTH OF THE INPUT FIELD
C INU - UNIT # TO OPEN THE MENU FILE ON
C*****
```

```
CHARACTER*(*) FNAME
CHARACTER*80 LINE(1),SCREEN(24),LINE1
CHARACTER*3 ID
```

```
INTEGER MN, INU, ROW, COL, RC(N,3), STATE, IS, INX, L, MIND(50)
```

```
LOGICAL MREAD
```

```
COMMON/MENUS/ MREAD
```

```
C OPEN THE MENU FILE DIRECT ACCESS (INITIALLY SEQUENTIAL)
C OPEN (INU,FILE=FNAME,ERR=100,ACCESS='DIRECT',RECL=80)
```

```
C FORM A DIRECT ACCESS ARRAY ON THE FIRST TIME THIS ROUTINE IS CALLED.
C IF (MREAD) THEN
```

```
C SET THE INDICATOR SO NO MORE CONVERSIONS TAKE PLACE
C MREAD=.FALSE.
```

```
C SET THE INDEX AND RECORD TO 1
C ICNT=1
C IREC=1
```

```
C READ IN THE MENU HEADERS (*****)
C READ(INU,REC=IREC,ERR=200) LINE(1)
```

```
C CONVERT ### TO AN INTEGER VALUE
C READ(LINE(1)(3:5),'(I3)') MIND(ICNT)
```

```
C INCREMENT THE ARRAY INDEX AND THE RECORD COUNTER, THE NEXT
C HEADER RECORD IS 24 AWAY
C ICNT= ICNT+1
C IREC= IREC+24
C GO TO 5
C ENDIF
```

```
C DETERMINE WHICH MENU HAS BEEN SELECTED AND DETERMINE RECORD NUMBER
200 DO 10 I=1,50,1
C IF(MIND(I) .EQ. MN) GOTO 12
10 CONTINUE
```

```

C      FORM THE RECORD NUMBER AND CLOSE THE MENU FILE
12     IREC= (I-1)*24 +1
        CLOSE(INU)

C      OPEN THE MENU FILE AND CLEAR THE SCREEN
        OPEN (INU, FILE=FNAME, ERR=100, ACCESS='DIRECT', RECL=80)
        CALL CLEAR(7,0)

C      PROCESS THE MENU, DISPLAY TEXT ON THE SCREEN AND FORM THE INPUT FIELDS
        DO 15 ROW=1,23,1
            IREC= IREC+1
            STATE=1
            READ(INU, REC=IREC, ERR=100) LINE(1)

C      SET THE COLUMN COUNTER TO THE FIRST COLUMN
        COL=1
20     JJ= COL+1

C      STATE 1 IS THE INITIAL STATE IS IS REMAINED IN UNTIL A NON BLANK
C      CHARACTER IS ENCOUNTERED
        IF ( STATE.EQ.1 ) THEN
            IF (LINE(1)(COL:COL) .NE. ' ') THEN
                IF (LINE(1)(COL:JJ) .EQ. 'X') THEN
                    STATE=2
                ELSE
                    STATE=3
                    IS=COL
                ENDIF
            ENDIF
        ENDIF

        IF (STATE.EQ.3 ) THEN
            IF (LINE(1)(COL:COL) .EQ. 'X') THEN
                IF (LINE(1)(JJ:JJ) .EQ. 'X') THEN
                    SCREEN(ROW)(IS:COL-1)=LINE(1)(IS:COL-1)
                    J1=COL-1
                    LINE1=LINE(1)
                    CALL MENUADR(LINE1(IS:J1), ROW, IS, 1, 0, 1, 1)
                    STATE=2
                ENDIF
            ENDIF
        ENDIF

        IF (STATE .EQ. 2) THEN
            IF (LINE(1)(COL+4:COL+4) .NE. 'X') GOTO 100
            IF (LINE(1)(COL+7:COL+8) .NE. 'X') GOTO 100
            READ(LINE(1)(COL+2:COL+3), '(I2)', ERR=100) INX
            READ(LINE(1)(COL+5:COL+6), '(I2)', ERR=100) L
            RC(INX,1)=ROW
            RC(INX,2)=COL
            RC(INX,3)=L
            STATE=1
            COL=COL+8
        ENDIF

C      INCREMENT THE COLUMN COUNTER AND TEST IF THIS IS THE LAST COLUMN
        COL=COL+1
        IF (COL .LE. 80) GOTO 20

C      NO MORE DATA LEFT IN THE RECORD, BUT THERE MAY BE DATA IN LINE(1)
C      THAT NEEDS TO BE OUTPUT

```

```

        IF (STATE .EQ. 3) THEN
            SCREEN(ROW) (IS:80)=LINE(1) (IS:80)
            LINE1=LINE(1)
            CALL MENUWR(LINE1 (IS:80), ROW, IS, 1, 0, 1, 1)
        ENDIF

15      CONTINUE

C      CLOSE THE FILE FROM EITHER NORMAL OR ABNORMAL TERMINATION
100     CLOSE (INU)
        RETURN
        END
        SUBROUTINE MENUWR(RC,N, IS, IE, TEXT, ICLR, ICOLOR, S)

C*****
C  THIS SUBROUTINE DISPLAYS AN ARRAY OF TEXT ON THE SCREEN AT THE ROW AND
C  COLUMN POSITIONS AND LENGTHS AS GIVEN IN THE RC ARRAY.  THE ADDRESSING INTO
C  THE TEXT ARRAY IS EITHER RELATIVE OR ABSOLUTE
C  VARIABLES PASSED:
C
C      RC      -  ARRAY TO HOLD THE INPUT FIELD DATA
C                RC(N,1) - ROW POSITION OF INPUT FIELD
C                RC(N,2) - COLUMN POSITION OF INPUT FIELD
C                RC(N,3) - LENGTH OF INPUT FIELD
C      N      -  SIZE OF THE RC ARRAY IN THE FIRST INDEX
C      IS     -  STARTING INDEX (RELATIVE) INTO THE TEXT ARRAY
C      IE     -  ENDING INDEX (RELATIVE) INTO THE TEXT ARRAY
C      TEXT   -  ARRAY OF THE CHARACTER DATA TO BE DISPLAYED ON THE SCREEN
C      ICLR   -  FLAG TO INDICATE WHETHER THE SCREEN IS CLEARED
C                1 ==> CLEAR THE SCREEN
C                ==> NOT TO CLEAR THE SCREEN
C      ICOLOR -  COLOR OF THE DISPLAYED TEXT
C      S      -  STATUS ARRAY USED TO PASS PARAMETERS TO MENUWR
C                S(1) - USED TO INITIALIZE PAGE ARRAY AND WRITE PAGE ARRAY
C                S(2) - USED TO PASS UNIT # OF ARCHIVE FILE
C                S(3) - INDICATE RELATIVE OR ABSOLUTE ADDRESSING IN TEXT ARRAY
C                      0 ==> ABSOLUTE ADDRESSING
C                      ==> RELATIVE ADDRESSING
C*****

        CHARACTER(*) TEXT(1)

        INTEGER      IS, IE, ICLR, ICOLOR, RC(N,3), K, N, S(1)

C      IF ICLR = 1 THEN CLEAR THE SCREEN
        IF (ICLR .EQ. 1) CALL CLEAR(7,0)

C      OUTPUT THE LINES OF TEXT
        IF (S(3) .EQ. 0) THEN

C          RELATIVE ADDRESSING MODE TO BE USED
            DO 5 K=IS, IE, 1
                CALL MENUWR(TEXT(K) (1:RC(K,3)), RC(K,1), RC(K,2), ICOLOR, S(1),
*                      S(2), 1)
5          CONTINUE

            ELSE

C          RELATIVE ADDRESSING MODE IS TO BE USED
            L=IE-IS+1
            DO 10 K=1, L, 1
                J=IS+(K-1)

```

```

      CALL MENUUDR(TEXT(K)(1:RC(J,3)), RC(J,1), RC(J,2), ICOLOR, S(1),
*          S(2),1)
10      CONTINUE
      ENDIF

      RETURN
      END
      SUBROUTINE MENURD(RC,N,IS,IE,TEXT,ITT)

C*****
C  THIS SUBROUTINE DISPLAYS DATA IN THE INPUT FIELDS AND INPUTS DATA FROM THE
C  INPUT FIELDS
C    VARIABLES PASSED:
C
C    RC  -  ARRAY THAT HOLDS THE INPUT FIELD DATA
C           RC(N,1) - ROW POSITION OF THE INPUT FIELD
C           RC(N,2) - COLUMN POSITION OF THE INPUT FIELD
C           RC(N,3) - LENGTH OF THE INPUT FIELD
C    N    -  SIZE OF FIRST POSITION OF THE RC ARRAY
C    IS   -  STARTING INDEX (RELATIVE) INTO THE TEXT ARRAY
C    IE   -  ENDING INDEX (RELATIVE) INTO THE TEXT ARRAY
C    TEXT -  ARRAY TO HOLD THE INPUTTED DATA
C    ITT  -  FLAG TO INDICATE RELATIVE OR ABSOLUTE ADDRESSING
C           >= 0 ==> ABSOLUTE ADDRESSING
C           ==> RELATIVE ADDRESSING
C*****

      CHARACTER*(*) TEXT(1)
      CHARACTER*80 BK,UN,INPUT
      CHARACTER*5 FMT

      INTEGER IS,IE,L,K,ITR,J,BNDX,ID,N,M,RC(N,3),ITT

C  INITIALIZE THE VARIABLES BK TO BLANKS AND UN TO UNDERSCORES
      IUN= 95
      DO 3 I=1,80,1
        BK(I:I)= ' '
        UN(I:I)= CHAR(IUN)
3      CONTINUE

C  SET THE INTERACTIVE TERMINAL READ UNIT TO 0 SINCE 0 AND -0 ARE THE SAME
      ITR=0

C  CHECK FOR THE TYPE OF ADDRESSING, RELATIVE OR ABSOLUTE
      IF (ITT .GE. 0) THEN

C  ABSOLUTE ADDRESSING SELECTED IN THE TEXT ARRAY
      DO 5 K=IS,IE,1

C      BUILD THE FORMAT STATEMENT
      FMT='(AXX)'
      L=RC(K,3)
      WRITE(FMT(3:4),'(I2)') L

C      SET UP THE LINE FOR INPUT, INITIALLY DISPLAY UNDERSCORES
      CALL MENUUDR(UN(1:L),RC(K,1),RC(K,2),1,0,1,1)

C      SEE IF THERE IS ALREADY DATA STORED FOR THIS INPUT FIELD
      IF (TEXT(K)(1:L) .EQ. BK(1:L)) THEN

```

```

C      SINCE NO DATA ALREADY STORED, HIGHLIGHT THE FIRST UNDERSCORE
      CALL MENU DR(UN(1:1), RC(K, 1), RC(K, 2), 7, 0, 1, 1)
      ID=1
C
C      ELSE
      DATA ALREADY STORED, WRITE IT OUT INTO THE UNDERSCORES FOR AS
      MANY NON BLANKS IT CONTAINS
      J=BNDX(TEXT(K), L)
      CALL MENU DR(TEXT(K) (1:J), RC(K, 1), RC(K, 2), 1, 0, 1, 1)
      CALL MENU DR(TEXT(K) (1:1), RC(K, 1), RC(K, 2), 7, 0, 1, 1)
      ID=0
C
C      ENDIF

C      MOVE THE CURSOR TO THE FIRST POSITION OF THE INPUT FIELD IN REVERSE
      VIDEO
      CALL MENU DR('~', RC(K, 1), RC(K, 2), 7, 0, 1, 1)

C      READ IN THE RESPONSE
      READ(ITR, '(A80)') INPUT

C      CHECK TO SEE IF THE NEW INPUT DATA IS BLANK OR NOT
      IF (INPUT(1:L) .NE. BK(1:L)) THEN

C      INPUT WAS NOT BLANK, SO DISPLAY IT AND MOVE ON TO THE NEXT INPUT
      FIELD
      TEXT(K) (1:L)=INPUT(1:L)
      CALL MENU DR(TEXT(K) (1:L), RC(K, 1), RC(K, 2), 1, 0, 1, 1)
      ELSE

C      INPUT WAS BLANK, DETERMINE IF THE ORIGINAL DATA WAS BLANK OR NOT
      IF BLANK, THEN BLANK THE INPUT FIELD, OTHERWISE DISPLAY THE OLD
      DATA
      IF (ID .EQ. 1) CALL MENU DR(BK(1:L), RC(K, 1), RC(K, 2), 1, 0, 1, 1)
      IF (ID .EQ. 0) CALL MENU DR(TEXT(K) (1:L), RC(K, 1), RC(K, 2), 1, 0, 1, 1)
      ENDIF

S      CONTINUE

C      ELSE
      RELATIVE ADDRESSING WAS SELECTED, OTHERWISE SAME AS ABOVE
      LL=IE-IS+1
      DO 10 J=1, LL, 1
        K=IS+(J-1)
C      BUILD THE FORMAT STATEMENT
        FMT='(AXX)'
        L=RC(K, 3)
        WRITE(FMT(3:4), '(I2)') L

C      SET UP THE LINE FOR INPUT
        CALL MENU DR(UN(1:L), RC(K, 1), RC(K, 2), 1, 0, 1, 1)
        IF (TEXT(J) (1:L) .EQ. BK(1:L)) THEN
          CALL MENU DR(UN(1:1), RC(K, 1), RC(K, 2), 7, 0, 1, 1)
          ID=1
        ELSE
          JJ=BNDX(TEXT(J), L)
          CALL MENU DR(TEXT(J) (1:JJ), RC(K, 1), RC(K, 2), 1, 0, 1, 1)
          CALL MENU DR(TEXT(J) (1:1), RC(K, 1), RC(K, 2), 7, 0, 1, 1)
          ID=0
        ENDIF
      CALL MENU DR('~', RC(K, 1), RC(K, 2), 7, 0, 1, 1)

```

```

C      READ IN THE RESPONSE
      READ(ITR,'(A80)') INPUT
      IF (INPUT(1:L) .NE. BK(1:L)) THEN
        TEXT(J)(1:L)=INPUT(1:L)
        CALL MENUUDR(TEXT(J)(1:L),RC(K,1),RC(K,2),1,0,1,1)
      ELSE
        IF (ID .EQ. 1) CALL MENUUDR(BK(1:L),RC(K,1),RC(K,2),1,0,1,1)
        IF (ID .EQ. 0) CALL MENUUDR(TEXT(J)(1:L),RC(K,1),RC(K,2),1,0,1,1)
      ENDIF

```

10 CONTINUE

ENDIF

```

RETURN
END
SUBROUTINE MENUUDR(IN,OUT,N,FMT,EFLAG)

```

```

C*****
C  THIS SUBROUTINE CONVERTS THE CHARACTER DATA IN THE ARRAY IN TO REAL DATA
C  DATA IN THE OUT ARRAY
C    VARIABLES PASSED:
C
C    IN   -  ARRAY OF TEXT TO BE CONVERTED TO REAL DATA
C    N    -  SIZE OF THE TEXT ARRAY, AND THE NUMBER OF ELEMENTS TO CONVERT
C    FMT  -  NOT USED
C
C    VARIABLES RETURNED:
C
C    OUT   -  ARRAY OF CONVERTED CHARACTER DATA (REAL)
C    EFLAG -  FLAG TO INDICATE WHETHER ANY ERRORS OCCURED IN CONVERSION
C             0 ==> NO CONVERSION ERRORS
C             I ==> INDEX OF THE DATA ITEM A CONVERSION ERROR OCCURRED
C*****

```

```

      CHARACTER*(*)  IN(N),FMT
      CHARACTER*80   TEMP

      INTEGER        N,EFLAG,I,L

      REAL            OUT(N)

```

```

C    INITIALIZE THE ERROR FLAG TO NO ERRORS
      EFLAG=0

C    DETERMINE THE LENGTH OF THE CHARACTER DATA
      L=LEN(IN(1))

C    CONVERT THE CHARACTER DATA TO REAL DATA
      DO 5 I=1,N,1
        TEMP(1:80)=' '

C        IF NO DECIMAL POINT PRESENT THEN PUT ONE IN
        IF (INDEX(IN(I),'.') .EQ. 0) THEN
          TEMP(L+1:L+1)='.'
          TEMP(1:L)=IN(I)
        ELSE
          TEMP(1:L)=IN(I)
        ENDIF

```

```

C      CONVERT THE CHARACTER DATA TO REAL
      READ(TEMP(1:80),'(F80.1)',ERR=10) OUT(I)
5      CONTINUE
      RETURN

C      SET THE ERROR FLAG TO THE INDEX WITH THE CONVERSION ERROR
10     EFLAG=I
      RETURN
      END
      SUBROUTINE MENU DR (STRING, ROW, COLUMN, FG, BG, H, W)

C*****
C  THIS SUBROUTINE MOVES THE CURSOR TO THE ROW AND COLUMN POSITON OF THE SCREEN
C  AS DESCRIBED BY ROW AND COLUMN.  ROW=1,2,3,...,24 AND COLUMN=1,2,3,...,80
C  VARIABLES PASSED:
C
C  STRING      - TEXT STRING TO BE DISPLAYED ON THE SCREEN
C                ~ ==> JUST MOVE TO POSITION DO NOT DISPLAY
C  ROW         - ROW TO MOVE TO ON THE SCREEN
C  COLUMN      - COLUMN TO MOVE TO ON THE SCREEN
C  FG          - COLOR TO DISPLAY THE TEXT IN ON THE SCREEN
C                6 OR 7 ==> REVERSE VIDEO
C                OTHER ==> NO REVERSE VIDEO
C  BG          - FLAG TO TELL MENU DR TO STORE OR OUTPUT DATA
C                -1 ==> INITIALIZE PAGE ARRAY AND STORE TEXT IN IT
C                -2 ==> OUTPUT THE PAGE ARRAY TO UNIT H
C  H           - UNIT # THAT ARCHIVE FILE IS OPEN UNDER
C  W           - NOT USED
C*****

      CHARACTER*(*)  STRING
      CHARACTER*80   PAGE(22)
      CHARACTER*4    CURSOR
      INTEGER        L, FG, BG, H, W, ROW, COLUMN

C      INITIALIZE THE PAGE ARRAY AND START STORING TEXT
      IF (BG .EQ. -1) THEN
          DO 10 J=1,22,1
              PAGE(J)(1:80)=' '
10          CONTINUE
          ENDIF

C      DETERMINE THE STRING LENGTH
      L=LEN (STRING)

C      STORE THE TEXT IN PAGE IF THERE IS TEXT TO STORE
      IF ((L .NE. 0).AND.(ROW .LT. 23))
      *      PAGE(ROW) (COLUMN: COLUMN+L-1)=STRING(1:L)

C      WRITE THE PAGE FILE TO THE ARCHIVE FILE ON UNIT H
      IF (BG .EQ. -2) THEN
          DO 15 J=1,22,1
              WRITE(H,'(A80)') PAGE(J)
15          CONTINUE
          ENDIF

C      SET THE CURSOR TO THE DESIRED POSITION BY SENDING ESC Y
      IESC=27
      CURSOR(1:1)=CHAR(IESC)

```

```
CURSOR(2:2)='Y'
CURSOR(3:3)=CHAR(ROW+31)
CURSOR(4:4)=CHAR(COLUMN+31)
```

```
C   MOVE THE CURSOR TO THE DESIRED POSITION
C   CALL TNOUA(CURSOR,4)
```

```
C   OUTPUT THE STRING IF IT IS NOT ~
C   IF (L .EQ. 1) THEN
C       IF (STRING .EQ. '~') RETURN
C   ENDIF
```

```
C   SET THE TERMINAL INTENSITY IN ZENITH MODE
C   ESC q ==> REVERSE VIDEO
C   ESC p ==> NO REVERSE VIDEO
CURSOR(2:2)='q'
IF ((FG .EQ. 7).OR. (FG .EQ. 6)) CURSOR(2:2)='p'
CALL TNOUA(CURSOR,2)
```

```
C   OUTPUT THE TEXT STRING THROUGH THE 68000 ASSEMBLER ROUTINE TNOUA
CALL TNOUA(STRING,L)
```

```
RETURN
END
SUBROUTINE CLEAR(FG,BG)
```

```
C*****
C   THIS SUBROUTINE CLEARS THE SCREEN BY TRANSMITTING AN ESC E IN ZENITH MODE
C   VARIABLES PASSED:
C
C   FG,BG - IGNORED IN THIS VERSION
C*****
```

```
CHARACTER*2  TEMP
INTEGER      FG,BG
```

```
IESC=27
TEMP(1:1)=CHAR(IESC)
TEMP(2:2)='E'
```

```
CALL TNOUA(TEMP,2)
```

```
RETURN
END
SUBROUTINE ONOFF(I)
```

```
C*****
C   THIS SUBROUTINE TURNS THE CURSOR EITHER ON OR OFF.  ESC x5 TURNS THE CURSOR
C   OFF, WHILE ESC y5 TURNS THE CURSOR ON.  WHEN TURNING THE CURSOR ON, BE SURE
C   TO EXIT REVERSE VIDEO MODE.
C   VARIABLES PASSED:
C
C   I - FLAG TO TURN THE CURSOR ON OR OFF
C       0 ==> CURSOR OFF
C       1 ==> CURSOR ON
C*****
```

```
CHARACTER*3  TEMP
INTEGER      I
```

```

C      TURN THE CURSOR ON OR OFF
      IESC=27
      TEMP(1:1)=CHAR(IESC)
      TEMP(2:3)='x5'
      IF (I .EQ. 1) TEMP(2:2)='y'
      CALL TNOUA(TEMP,3)

C      BE SURE TO EXIT REVERSE VIDEO MODE, SEND AN ESC q
      IF (I .EQ. 1) THEN
          TEMP(2:2)='q'
          CALL TNOUA(TEMP,2)
      ENDIF

```

```

      RETURN
      END
      SUBROUTINE MESS(I,R,C,L,CL)

```

```

C*****
C  THIS SUBROUTINE DISPLAY MESSAGE ON THE TERMINAL
C    VARIABLES PASSED:
C
C    I  -  MESSAGE INDEX
C    R  -  ROW ON THE SCREEN TO DISPLAY THE MESSAGE
C    C  -  COLUMN ON THE SCREEN TO DISPLAY THE MESSAGE
C    L  -  LENGTH OF THE MESSAGE TO BE DISPLAYED
C    CL -  COLOR TO DISPLAY THE MESSAGE
C          6 OR 7 ==> REVERSE VIDEO
C          OTHER  ==> NO REVERSE VIDEO
C*****

```

```

      CHARACTER*30  MES(25)

```

```

      INTEGER      I,R,C,L,CL

```

```

      MES(1)(1:30)='INVALID DATA'
      MES(2)(1:30)='ENTRY EXISTS'
      MES(3)(1:30)='ADDITION MADE'
      MES(4)(1:30)=' '
      MES(5)(1:30)='NOT FOUND'
      MES(11)(1:30)='INVALID COMMAND'
      MES(12)(1:30)='IN ARCHIVE MODE'
      MES(13)(1:30)='ARCHIVE MODE'
      MES(14)(1:30)='TOO MANY FILES'
      MES(15)(1:30)='HELP NOT AVAILABLE'
      MES(16)(1:30)='PRESS RETURN TO CONTINUE'
      MES(17)(1:30)='NO ARCHIVE FILES FOUND'
      MES(18)(1:30)='DATABASE GONE'
      MES(19)(1:30)='PRESS RETURN TO EXIT'
      MES(20)(1:30)='NO FILENAME'
      MES(21)(1:30)='99 FILES EXIST'
      MES(22)(1:30)='NO PROCEDURE FILES'

```

```

C      DISPLAY THE DESIRED MESSAGE
      CALL MENU DR(MES(I)(1:L),R,C,CL,0,1,1)

```

```

      RETURN
      END

```



```

$bigcode
$segment %tccmseg
      SUBROUTINE TCCM(ITR,AUNIT,AFLAG,SUNIT1,SUNIT2,SUNIT3,SUNIT4,
*          CUNIT,CFILE,SFILE1,SFILE2,SFILE3,SFILE4,
*          SMUNIT,SMFILE,MUNIT,MFILE)

C*****
C  THIS SUBROUTINE MANAGES THE DATA FOR THE TOXIC CORRIDOR CALCULATIONS.
C  VARIABLES PASSED:
C
C      ITR      -  INTERACTIVE TERMINAL READ UNIT
C      AUNIT    -  UNIT # ARCHIVE FILE IS OPEN ON
C      AFLAG    -  FLAG TO INDICATE WHETHER IN ARCHIVE MODE
C                  TRUE ==>  IN ARCHIVE MODE
C                  FALSE ==> NOT IN ARCHIVE MODE
C      SUNIT1   -  UNIT # TO OPEN SFILE1 ON
C      SUNIT2   -  UNIT # TO OPEN SFILE2 ON
C      SUNIT3   -  UNIT # TO OPEN SFILE3 ON
C      SUNIT4   -  UNIT # TO OPEN SFILE4 ON
C      CUNIT    -  UNIT # TO OPEN CFILE ON (NOT USED)
C      CFILE    -  CLIMATOLOGICAL DATABASE FILE NAME (NOT USED)
C      SFILE1   -  SUBSTANCE FILE NAME
C      SFILE2   -  SOURCE FILE NAME
C      SFILE3   -  POINTER FILE NAME
C      SFILE4   -  DATA FILE NAME
C      SMUNIT   -  UNIT # TO OPEN MENU FILE (SMFILE) ON
C      SMFILE   -  MENU FILE NAME
C      MUNIT    -  UNIT # TO OPEN MAP FILES ON
C      MFILE    -  SUPER DIRECTORY FILE FOR MAPS
C*****

      CHARACTER*1  INP,CMD(1)
      CHARACTER*3  TT
      CHARACTER*8  DATE8,TEMP0,CC(10)
      CHARACTER*7  CFILE,SFILE1,SFILE2,SFILE3,SFILE4,SMFILE,MFILE
      CHARACTER*40 SKEY(2)
      CHARACTER*80 BK80(1)

      INTEGER      ITR,I,SUNIT1,SUNIT2,SUNIT3,SUNIT4,AUNIT,
*          CUNIT,RC(12,3),SMUNIT,ST(3),ITT,EFLAG,MUNIT

      REAL          SDATA(14),TDATA(2),CDATA(4)

      LOGICAL       FLAG1,FLAG2,FLAG3,FLAG4,AFLAG

      DATA ST /0,0,0/
      DATA TDATA/-1,-1/

      BK80(1)(1:80)=' '
      INP=' '
      ITT=-1
      SKEY(1)(1:40)=' '
      SKEY(2)(1:40)=' '

100  DO 1 I=1,11,1
      CC(I)(1:8)=' '
1    CONTINUE

C    DISPLAY THE TCCM MENU
      CALL MENUSV(SMFILE,300,RC,12,SMUNIT)

```

```

C      DISPLAY THE SOURCE OR BLANK
      ST(3)=0
      CALL MENUWR(RC,12,2,2,SKEY,0,1,ST)

C      IF THE SOURCE WAS GIVEN AS A ? THEN THE USER WAS ALLOWED TO
C      CHOOSE FROM A DISPLAYED LIST. WHEN HE HAS MADE HIS CHOICE, THE
C      MAIN MENU IS DISPLAY ALONG WITH BOTH CHOICES AND THE USER IS
C      THEN PLACED AT THE SELECT OPTION LINE.
      IF (INP.EQ. '&') THEN
          CALL MENUWR(RC,12,1,1,SKEY,0,1,ST)
          GOTO 105
      ENDIF

C      INPUT THE SUBSTANCE AND THE SOURCE
      IST=1
5      CALL MENURD(RC,12,IST,2,SKEY,ITR)

C      CHECK TO SEE IF THE SUBSTANCE AND/OR THE SOURCE ARE BLANK OR *.
C      IF EITHER IS THEN GO BACK AND GET A NON-BLANK CHARACTER STRING
      IF ((SKEY(1).EQ. ' ').OR.(SKEY(2).EQ. ' ')) THEN
          IF (SKEY(2).EQ. ' ') IST=2
          IF (SKEY(1).EQ. ' ') IST=1
          GOTO 5
      ENDIF
      IF ((SKEY(1).EQ. '*').OR.(SKEY(2).EQ. '*')) THEN
          IF (SKEY(2).EQ. '*') IST=2
          IF (SKEY(1).EQ. '*') IST=1
          GOTO 5
      ENDIF

C      DETERMINE WHICH COMMAND THE USER HAS SELECTED AND EXECUTE THAT
C      OPTION PROVIDED THAT IT IS A VALID SELECTION.
105     CMD(1)=' '
      CALL MENURD(RC,12,11,11,CMD,ITT)

C      USER WISHES TO GO BACK TO THE SUBSTANCE
      IF (CMD(1).EQ. ' ') THEN
          IST=1
          ST(3)=1
          CALL MENUWR(RC,12,11,11,CMD,0,1,ST)
          CALL MESS(4,RC(12,1),RC(12,2),RC(12,3),1)
          GOTO 5
      ENDIF

C      USER WISHES TO USE THE MANUAL MODE OF THE TCCM
      IF (CMD(1).EQ. 'M') THEN
          INQUIRE (FILE=SF1,EXIST=FLAG1)
          INQUIRE (FILE=SF2,EXIST=FLAG2)
          INQUIRE (FILE=SF3,EXIST=FLAG3)
          INQUIRE (FILE=SF4,EXIST=FLAG4)
          IF ((.NOT. FLAG1).OR.(.NOT. FLAG2).OR.
              (.NOT. FLAG3).OR.(.NOT. FLAG4)) THEN
              CALL MESS(18,RC(12,1),RC(12,2),RC(12,3),6)
              GOTO 105
          ENDIF
          CALL TCCM1(ITR,AUNIT,AFLAG,SUNIT1,SUNIT2,SUNIT3,
              SUNIT4,CUNIT,CFILE,SF1,SF2,
              SF3,SF4,SKEY,SMUNIT,SMFILE,
              MUNIT,MFILE)

```

GOTO 100
ENDIF

C USER WISHES TO RETURN
IF (CMD(1) .EQ. 'X') RETURN

C INVALID COMMAND
IF (CMD(1) .NE. 'C') GOTO 105

C PROCESS THE TCCM DATA. THIS IS THE SECTION OF THE CODE THAT
C SEARCHES THE DATA BASE FOR THE DATA THAT IS STORED WITH THE
C GIVEN SUBSTANCE AND SOURCE. WHEN THE SUBSTANCE AND THE SOURCE
C ARE SPECIFIED, NO MATTER HOW THAT MAY BE, IT WILL EVENTUALLY BE
C PROCESSED THROUGH THIS SECTION OF CODE.
IF ((SKEY(1) .NE. '?').AND. (SKEY(2) .NE. '?')) THEN
EFLAG=0
INP='&'

C SEARCH THE SUBSTANCE-SOURCE DATABASE FOR THE DESIRED ENTRY
* CALL SSEAR(EFLAG, ITR, SUNIT1, SUNIT2, SUNIT3, SUNIT4, INP, SKEY,
SDATA, SFILE1, SFILE2, SFILE3, SFILE4, SMUNIT, SMFILE)

C EFLAG = 0 INDDICATES THAT THE VALUES WERE FOUND IN THE DATA BASE
IF (EFLAG .EQ. 0) THEN

C CONVERT THE REAL DATA TO ALPHA DATA
CALL MESS(4, RC(12, 1), RC(12, 2), RC(12, 3), 1)
WRITE(CC(3) (1:8), 25) SDATA(2)
WRITE(CC(4) (1:8), 25) SDATA(3)
WRITE(CC(5) (1:8), 25) SDATA(4)
WRITE(CC(6) (1:8), 25) SDATA(11)
25 FORMAT(F8.3)

C DISPLAY THE SUBSTANCE-SOURCE DATA
ST(3)=0
CALL MENUWR(RC, 12, 1, 2, SKEY, 0, 7, ST)
CALL MENUWR(RC, 12, 3, 6, CC, 0, 7, ST)
CALL MENUWR(BK80, 23, 1, 1, 0, 1, 1)
RC(11, 2)=30
CALL MENUWR('C(ONTINUE) OR X(RETURN) ==>', 23, 1, 1, 0, 1, 1)

C THIS IS WHERE THE I/O DIRVERS WILL INPUT THE TEMPERATURE DATA
DO 169 K=1, 10, 1
CC(K) (1:8)=' '
169 CONTINUE

C INPUT TEMPERATURE DATA
IST=7
170 CALL MENURD(RC, 12, IST, 10, CC, ITR)

C INPUT THE USER SELECTION
171 CMD(1)=' '
CALL MENURD(RC, 12, 11, 11, CMD, ITT)

C USERS WISHES TO RETURN
IF (CMD(1) .EQ. 'X') RETURN

C USER WISHES TO GO BACK TO THE SOURCE STRENGTH
IF (CMD(1) .EQ. ' ') THEN
IST=7

```

                                ST(3)=1
                                CALL MENUWR(RC,12,11,11,CMD,0,1,ST)
                                CALL MESS(4,RC(12,1),RC(12,2),RC(12,3),1)
                                GOTO 170
                                ENDIF

C      USER SELECTED INVALID INPUT
      IF (CMD(1) .NE. 'C') GOTO 171

C      CHECK FOR VALID CLIMO INPUT
      DO 168 K=1,4,1
        CC(K)=CC(K+6)
168    CONTINUE
      CALL MENUCK(CC,CDATA,4,'F8.3',IERR)

C      INVALID CLIMO DATA ENTERED
      IF (IERR .NE. 0) THEN
        ST(3)=1
        IST=IERR+6
        CMD(1)=' '
        CALL MENUWR(RC,12,11,11,CMD,0,1,ST)
        CALL MESS(1,RC(12,1),RC(12,2),RC(12,3),6)
        GOTO 170
      ENDIF

C      PROCESS AND DISPLAY THE DATA
180    CALL TCCM2(ITR,AUNIT,AFLAG,SKEY,SDATA,CDATA,TDATA,
      *      SMUNIT,SMFILE,MUNIT,MFILE)
      INP=' '
      GOTO 100
                                ENDIF

C      EFLAG = 10 INDICATES THAT ONE OF THE 4 REQUIRED FILES FOR THE
C      SUBSTANCE-SOURCE DATA BASE WAS NOT FOUND.
C      EFLAG = 11 INDICATES THAT THE SPECIFIED SUBSTANCE WAS NOT FOUND.
C      ANY OTHER EFLAG INDICATES THAT THE SPECIFIED SOURCE WAS NOT FOUND.
      IF (EFLAG .EQ. 10) THEN
        CALL MESS(18,RC(12,1),RC(12,2),RC(12,3),6)
        GOTO 105
      ENDIF
      IF (EFLAG .EQ. 11) THEN
        IST=1
      ELSE
        IST=2
      ENDIF

      ST(3)=1
      CMD(1)=' '
      CALL MENUWR(RC,12,11,11,CMD,0,1,ST)
      CALL MESS(5,RC(12,1),RC(12,2),RC(12,3),6)
      GOTO 5
                                ENDIF

C      ALLOW THE USER TO SEARCH THE DATA BASE
      FLAG2=.FALSE.
      IF (SKEY(1) .EQ. '?') THEN
        FLAG2=.TRUE.
        INQUIRE (FILE=SFILE1,EXIST=FLAG1)
        IF (.NOT. FLAG1) THEN

```

```

        CALL MESS(18, RC(12, 1), RC(12, 2), RC(12, 3), 6)
        GOTO 105
    ENDIF

C      CALL ROUTINE TO DISPLAY A LIST OF THE POSSIBLE SUBSTANCES
      CALL SBQST(SKEY, SUNIT1, SUNIT2, SFILE1, SFILE2, SMUNIT, SMFILE,
*          ITR, EFLAG)

C      NO SUBSTANCE WAS FOUND THAT MATCHED TO BE DISPLAYED
      IF (EFLAG .NE. 0) THEN
          ST(3)=1
          CMD(1)=' '
          CALL MENUWR(RC, 12, 11, 11, CMD, 0, 1, ST)
          CALL MESS(5, RC(12, 1), RC(12, 2), RC(12, 3), 6)
          IST=2
          GOTO 5
          ENDIF

      INP='&'

      ENDIF

C      USER INPUT A ? FOR THE SOURCE
      IF (SKEY(2) .NE. '?') GOTO 100

C      ROUTINE TO DISPLAY A LIST OF THE SOURCES THAT ARE POSSIBLE TO
C      CHOOSE FORM
      CALL SRCQST(SKEY, SUNIT1, SUNIT2, SFILE1, SFILE2, SMUNIT, SMFILE,
*          ITR, EFLAG)

C      NO SOURCES WERE FOUND THAT MATCHED TO BE DISPLAYED
      IF (EFLAG .NE. 0) THEN
          ST(3)=1
          CMD(1)=' '
          CALL MENUWR(RC, 12, 11, 11, CMD, 0, 1, ST)
          CALL MESS(5, RC(12, 1), RC(12, 2), RC(12, 3), 6)
          IST=1
          GOTO 5
          ENDIF

      INP='&'
      GOTO 100

      END
      SUBROUTINE TCCM1(ITR, AUNIT, AFLAG, SUNIT1, SUNIT2, SUNIT3, SUNIT4,
*          CUNIT, CFILE, SFILE1, SFILE2, SFILE3, SFILE4, SKEY,
*          SMUNIT, SMFILE, MUNIT, MFILE)

C*****
C  THIS SUBROUTINE ALLOWS THE USER TO MANUALLY ENTER 10, 30, 60 MIN SPEL'S AND
C  SOURCE STRENGTH AND Z FACTORS AND SPILL AREAS AND TEMPERATURES
C  VARIABLES PASSED:
C
C  SAME AS FOR TCCM
C*****

      CHARACTER*1  INP, CMD(1)
      CHARACTER*8  CC(14), EC(4)
      CHARACTER*7  CFILE, SFILE1, SFILE2, SFILE3, SFILE4, SMFILE, MFILE
      CHARACTER*40  SKEY(2)

      INTEGER      ITR, I, EFLAG, SUNIT1, SUNIT2, SUNIT3, SMUNIT, ST(3),
*          SUNIT4, AUNIT, CUNIT, RC(16, 3), ITT, MUNIT

```

```

REAL          CDATA(4), TDATA(2), TEMP(4)
REAL          SDATA(14)

LOGICAL       AFLAG

DATA ST/0,0,0/

ITT=-1
DO 5 I=1,14,1
  SDATA(I)= 0.0
  CC(I)(1:8)=' '
5  CONTINUE

C  CHECK TO SEE IF THE SUBSTANCE IS IN THE DATA BASE, IF IT IS
C  THEN THE GMW AND 10,30, AND 60 MIN PEL EXIST.
  INP='&'
  EFLAG=0

C  SEARCH THE SUBSTANCE-SOURCE DATA BASE FOR A MATCH
  CALL SSEAR(EFLAG, ITR, SUNIT1, SUNIT2, SUNIT3, SUNIT4, INP, SKEY,
*          SDATA, SFILE1, SFILE2, SFILE3, SFILE4, SMUNIT, SMFILE)

C  IF EFLAG = 11 THEN THE SUBSTANCE WAS NOT FOUND IN THE DATA BASE.
  IF (EFLAG .NE. 11) THEN
C  SUBSTANCE WAS FOUND IN THE DATA BASE
C  DISPLAY THE MENU AND THE DATA RETRIEVED FROM THE DATA BASE
    CALL MENUSV(SMFILE, 110, RC, 16, SMUNIT)
    ST(3)=0
    CALL MENUWR(RC, 16, 1, 2, SKEY, 0, 1, ST)
    WRITE(CC(3)(1:8), '(F8.3)') SDATA(2)
    WRITE(CC(4)(1:8), '(F8.3)') SDATA(3)
    WRITE(CC(5)(1:8), '(F8.3)') SDATA(4)
    CALL MENUWR(RC, 16, 3, 5, CC, 0, 1, ST)

C  EFLAG=12 ==> SOURCE NOT FOUND AND NO SOURCE STRENGTH AVAILABLE
    IF (EFLAG .NE. 12) WRITE(CC(6)(1:8), '(F8.3)') SDATA(11)

C  INPUT THE SOURCE STRENGTH FROM THE USER
125 CALL MENURD(RC, 16, 6, 6, CC, ITR)
    IF (CC(6) .EQ. ' ') GOTO 125

C  CONVERT THE CHARACTER DATA TO REAL DATA
    EC(1)=CC(6)
    CALL MENUCK(EC, TEMP, 1, '(F8.3)', IERR)

C  SINCE NO ERROR OCCURRED IN THE CONVERSION ==> THE USER IS NOT GOING
C  TO USE THE Z FACTOR METHOD
    IF (IERR .EQ. 0) THEN
      SDATA(11)=TEMP(1)
      TDATA(1)=-1
      TDATA(2)=-1
      CC(7)(1:8)=' '
      CC(8)(1:8)=' '
      CC(9)(1:8)=' '
      ST(3)=0
      CALL MENUWR(RC, 16, 7, 9, CC, 0, 1, ST)
      IST=10
      GOTO 170
    ENDIF

```

```

C      USER IS GOING TO USE THE Z FACTOR METHODD ==> ZFACTOR AND SPILL AREA
C      AND SPILL TEMPERATURE TO BE INPUT
      CC(6)(1:8)=' '
      WRITE(CC(7)(1:8),'(F8.3)') SDATA(5)
      ST(3)=0

C      DISPLAY THE Z FACTOR STORED IN THE DATABASE
      CALL MENUWR(RC,16,6,7,CC,0,1,ST)
      SDATA(11)=-1.0

C      INPUT THE SPILL PARAMETERS
      IST=7
130    CALL MENURD(RC,16,IST,9,CC,ITR)

C      INPUT THE TEMPERATURE DATA
      IST=10
170    CALL MENURD(RC,16,IST,13,CC,ITR)

C      SCANNING THE COMMAND INPUT LINE
160    CMD(1)=' '
      CALL MENURD(RC,16,14,14,CMD,ITT)

C      USER SELECTED TO RETURN
      IF (CMD(1) .EQ. 'X') RETURN

C      USER SELECTED TO GO BACK TO THE SOURCE STRENGTH INPUT LINE
      IF (CMD(1) .EQ. ' ') THEN
          ST(3)=1
          CALL MESS(4,RC(15,1),RC(15,2),RC(15,3),1)
          CALL MENUWR(RC,16,14,14,CMD,0,1,ST)
          GOTO 125
      ENDIF

C      INVALID INPUT
      IF (CMD(1) .NE. 'C') GOTO 160

C      CHECK THE SOURCE STRENGTH OR Z FACTOR DATA TO BE VALID
      IF (SDATA(11) .NE. -1.) THEN
          IF (SDATA(11) .LE. 0) THEN
              CMD(1)=' '
              ST(3)=1
              CALL MENUWR(RC,16,14,14,CMD,0,1,ST)
              CALL MESS(1,RC(15,1),RC(15,2),RC(15,3),7)
              GOTO 125
          ENDIF
      ELSE
C      CONVERT THE CHARACTER DATA TO REAL DATA
          EC(1)=CC(7)
          EC(2)=CC(8)
          EC(3)=CC(9)
          CALL MENUCK(EC,TEMP,3,'F(8.3)',IERR)

C      ERROR OCCURRED IN THE CONVERSION PROCESS
          IF (IERR .NE. 0) THEN
              CMD(1)=' '
              ST(3)=1
              CALL MENUWR(RC,16,14,14,CMD,0,1,ST)
              CALL MESS(1,RC(15,1),RC(15,2),RC(15,3),7)
              IST=IERR+6
          
```

```

                                GOTO 130
                                ENDIF
DO 180 K=1,3,1
  IF (TEMP(K) .LE. 0) THEN
    CMD(1)=' '
    ST(3)=1
    CALL MENUWR(RC,16,14,14,CMD,0,1,ST)
    CALL MESS(1,RC(15,1),RC(15,2),RC(15,3),7)
    IST=K+6
    GOTO 130
                                ENDIF
180      CONTINUE

C      PLACE Z FACTOR DATA INTO THE PROPER PLACES
SDATA(5)=TEMP(1)
TDATA(1)=TEMP(2)
TDATA(2)=TEMP(3)
                                ENDIF

C      CHECK FOR VALID CLIMQ INPUT
DO 175 I=1,4,1
  EC(I)=CC(I+9)
175  CONTINUE
CALL MENUCK(EC,CDATA,4,'(F8.3)',IERR)

C      ERROR OCCURRED IN THE CONVERSION PROCESS
IF (IERR .NE. 0) THEN
  CMD(1)=' '
  ST(3)=1
  CALL MENUWR(RC,16,14,14,CMD,0,1,ST)
  CALL MESS(1,RC(15,1),RC(15,2),RC(15,3),7)
  IST=IERR+9
  GOTO 170
ENDIF

                                ELSE
C      THE SUBSTANCE IS NOT IN THE DATA BASE
C      DISPLAY THE MENU AND THE SUBSTANCE AND THE SOURCE
CALL MENUSV(SMFILE,111,RC,16,SMUNIT)
ST(3)=0
CALL MENUWR(RC,16,1,2,SKEY,0,1,ST)

C      INPUT THE 10,30, AND 60 MIN SPELS
IST=3
220  CALL MENURD(RC,16,IST,6,CC,ITR)

C      INPUT THE SOURCE STRENGTH
225  CALL MENURD(RC,16,7,7,CC,ITR)
IF (CC(7) .EQ. ' ') GOTO 225

C      CONVERT THE CHARACTER DATA TO REAL DATA
EC(1)=CC(7)
CALL MENUCK(EC,TEMP,1,'(F8.3)',IERR)

C      IF NO ERROR OCCURRED ==> USER WILL NOT USE Z FACTOR METHOD
IF (IERR .EQ. 0) THEN
  SDATA(11)=TEMP(1)
  SDATA(5)=0.
  TDATA(1)=-1
  TDATA(2)=-1

```

```

                                CC(8)(1:8)=' '
                                CC(9)(1:8)=' '
                                CC(10)(1:8)=' '
                                ST(3)=0
                                CALL MENUWR(RC, 16, 8, 10, CC, 0, 1, ST)
                                IST=11
                                GOTO 270
                                ENDIF

C      USER WILL USE THE Z FACTOR METHOD
      CC(7)(1:8)=' '
      ST(3)=0
      CALL MENUWR(RC, 16, 7, 7, CC, 0, 1, ST)
      SDATA(11)=-1.0

C      INPUT THE Z FACTOR AND SPILL PARAMETERS
      IST=8
230    CALL MENURD(RC, 16, IST, 10, CC, ITR)

C      INPUT THE TEMPERATURE DATA
      IST=11
270    CALL MENURD(RC, 16, IST, 14, CC, ITR)

C      SCANNING THE COMMAND INPUT LINE
260    CMD(1)=' '
      CALL MENURD(RC, 16, 15, 15, CMD, ITT)

C      USER SELECTED TO RETURN
      IF (CMD(1) .EQ. 'X') RETURN

C      USER SELECTED TO GO BACK TO THE GMW, 10, 30, 60 SPEL LINES
      IF (CMD(1) .EQ. ' ') THEN
          ST(3)=1
          CALL MESS(4, RC(16, 1), RC(16, 2), RC(16, 3), 1)
          CALL MENUWR(RC, 16, 15, 15, CMD, 0, 1, ST)
          IST=3
          GOTO 220
          ENDIF

C      INVALID COMMAND
      IF (CMD(1) .NE. 'C') GOTO 260

C      CHECK THE GMW, 10, 30 AND 60 MIN PEL'S
      EC(1)=CC(3)
      EC(2)=CC(4)
      EC(3)=CC(5)
      EC(4)=CC(6)
      CALL MENUCK(EC, TEMP, 4, ' (F8.3) ', IERR)

C      ERROR IN THE CONVERSION PROCESS
      IF (IERR .NE. 0) THEN
          CMD(1)=' '
          ST(3)=1
          CALL MENUWR(RC, 16, 15, 15, CMD, 0, 1, ST)
          CALL MESS(1, RC(16, 1), RC(16, 2), RC(16, 3), 7)
          IST=IERR+2
          GOTO 220
          ENDIF
      DO 285 K=1, 4, 1
          IF (TEMP(K) .LE. 0) THEN

```

```

        CMD(1)=' '
        ST(3)=1
        CALL MENUWR(RC, 16, 15, 15, CMD, 0, 1, ST)
        CALL MESS(1, RC(16, 1), RC(16, 2), RC(16, 3), 7)
        IST=K+2
        GOTO 220
                                ENDIF
285      SDATA(K)=TEMP(K)
      CONTINUE
C      CHECK THE SOURCE STRENGTH AND Z FACTOR
      IF (SDATA(11) .NE. -1.) THEN
        IF (SDATA(11) .LE. 0) THEN
          CMD(1)=' '
          ST(3)=1
          CALL MENUWR(RC, 16, 15, 15, CMD, 0, 1, ST)
          CALL MESS(1, RC(16, 1), RC(16, 2), RC(16, 3), 7)
          GOTO 225
                                ENDIF
        ELSE
          EC(1)=CC(8)
          EC(2)=CC(9)
          EC(3)=CC(10)
          CALL MENUCK(EC, TEMP, 3, 'F(8.3)', IERR)
C      ERROR IN THE CONVERSION PROCESS
          IF (IERR .NE. 0) THEN
            CMD(1)=' '
            ST(3)=1
            CALL MENUWR(RC, 16, 15, 15, CMD, 0, 1, ST)
            CALL MESS(1, RC(16, 1), RC(16, 2), RC(16, 3), 7)
            IST=IERR+7
            GOTO 230
                                ENDIF
          DO 280 K=1, 3, 1
            IF (TEMP(K) .LE. 0) THEN
              CMD(1)=' '
              ST(3)=1
              CALL MENUWR(RC, 16, 15, 15, CMD, 0, 1, ST)
              CALL MESS(1, RC(16, 1), RC(16, 2), RC(16, 3), 7)
              IST=K+7
              GOTO 230
                                ENDIF
          280      CONTINUE
C      PLACE Z FACTOR DATA INTO THE PROPER PLACES
          SDATA(5)=TEMP(1)
          TDATA(1)=TEMP(2)
          TDATA(2)=TEMP(3)
                                ENDIF
C      CHECK FOR VALID CLIMO INPUT
          DO 275 I=1, 4, 1
            EC(I)=CC(I+10)
275      CONTINUE
          CALL MENUCK(EC, CDATA, 4, 'F(8.3)', IERR)
C      ERROR IN CONVERSION PROCESS
          IF (IERR .NE. 0) THEN
            CMD(1)=' '

```

```

                                ST(3)=1
                                CALL MENUWR(RC, 16, 15, 15, CMD, 0, 1, ST)
                                CALL MESS(1, RC(16, 1), RC(16, 2), RC(16, 3), 7)
                                IST=IERR+10
                                GOTO 270
                                ENDIF
                                ENDIF
C   MAKE THE TOXIC CORRIDOR CALCULATION
    CALL TCCM2(ITR, AUNIT, AFLAG, SKEY, SDATA, CDATA, TDATA, SMUNIT, SMFILE,
*       MUNIT, MFILE)
    RETURN
    END
    SUBROUTINE TCCM2(ITR, AUNIT, AFLAG, SKEY, SDATA, CDATA, TDATA,
-       SMUNIT, SMFILE, MUNIT, MFILE)
C
C   ARGUMENT VARIABLE TYPES
C
    INTEGER      ITR, AUNIT, SMUNIT, MUNIT
    REAL          SDATA(1), CDATA(1), TDATA(1)
    CHARACTER*40  SKEY(1)
    CHARACTER*7   SMFILE, MFILE
    LOGICAL       AFLAG
C
C   PROGRAM VARIABLE TYPES
C
    REAL          PF, Q, GMW, PEL(3), WINDS,
-               CORWID, DELTAT, SIGTH, DIST(3,2)
C   EXTERNAL FUNCTIONS
    REAL          OCEANT
C
C   INITIALIZE ARRAYS AND VARIABLES
C
    DO 10 I=1, 3, 1
        PEL(I)= -1.0
        IF (SDATA(I+1) .GT. 0.0) PEL(I)= SDATA(I+1)
    DO 20 J=1, 2, 1
        DIST(I, J)= -1.0
20    CONTINUE
10    CONTINUE
C
C   COMPUTE CORRIDOR BASED ON A 90% PROBABILITY LEVEL(PL):  FACTOR(PF)= 1.63
C
    PF= 1.63
C
C   DEFINE VARIABLES
C
    GMW= SDATA(1)
    WINDS= CDATA(1)
    SIGTH= CDATA(3)
    DELTAT= CDATA(4)
    Q= SDATA(11)
C
C   IF Q=-1, THE SOURCE STRENGTH MUST BE COMPUTED VIA THE SPILL EQUATION
C
    IF( Q.EQ.-1. ) THEN
        A= TDATA(1)
        TP= TDATA(2)
        Z= SDATA(5)
        Q= 1.66E-04 * (WINDS**.75) * A

```

```

      * ( 1.0 + ( 4.3E-03 *(TP**2.0) ) ) * Z
      SDATA(11) = -Q
      ENDIF
C
C COMPUTE CORRIDOR LENGTHS
C
      DO 30 I=1,3,1
        IF (PEL(I) .GT. 0.) THEN
          DIST(I,1) = OCEANT(PF,GMW,PEL(I),Q,DELTAT)
          IF (SIGTH .GT. 0.0) THEN
            DIST(I,2) = OCEANS(PF,GMW,PEL(I),Q,
                               SIGTH,DELTAT)
          ENDIF
        ENDIF
      30 CONTINUE
C
C COMPUTE CORRIDOR WIDTHS
C
      IF {WINDS.LE.3.0} THEN CORWID= 360.0
      GO TO 35
      IF {SIGTH.GT.0.0} THEN CORWID= 6.0 * SIGTH
      GO TO 35
      IF {WINDS.GT.10.0} THEN CORWID= 60.0
      ELSE CORWID= 90.0
      ENDIF
35 CONTINUE
C
      CALL TCDISP( ITR,AUNIT,AFLAG,SKEY,SDATA,CDATA,TDATA,
                  CORWID,DIST,SMUNIT,SMFILE,MUNIT,MFILE)
C
C
      RETURN
      END
      REAL FUNCTION OCEANT(PF,GMW,GLC,Q,DELTAT)
C
C COMPUTE CORRIDOR LENGTH BASED ON THE DELTA-T VERSION OF THE OCEAN
C BREEZE/DRY GULCH EQUATION
C
      REAL PF,GMW,GLC,Q,DELTAT
C
C CHECK FOR VALID INPUT
C
      IF( (GMW .LE. 0.) .OR. (Q.LE.0.) .OR. (PF.LE.0.) ) RETURN
C
      OCEANT= PF* ( 3.28
      .           *((29.75/GMW)**0.513)
      .           *((GLC/Q)**(-0.513))
      .           *((DELTAT+10)**2.53) )
C
      RETURN
      END
      REAL FUNCTION OCEANS(PF,GMW,GLC,Q,SIGTH,DELTAT)
C
C COMPUTE CORRIDOR LENGTH BASED ON THE SIGMA THETA VERSION OF THE OCEAN
C BREEZE/DRY GULCH EQUATION
C
      REAL PF,GMW,GLC,Q,SIGTH,DELTAT
C
C CHECK FOR VALID INPUT

```

```

C      IF( (GMW .LE. 0.) .OR. (Q.LE.0.) .OR. (PF.LE.0.) ) RETURN
C
C      OCEANS= PF* ( 3.28
C          *((357.0/GMW)**0.510)
C          *((GLC/Q)**(-0.510))
C          *(SIGTH**(-0.258))
C          *((DELTAT+10)**2.208)      )
C
C      RETURN
C      END
C      SUBROUTINE TCDISP(ITR,AUNIT,AFLAG,SKEY,SDATA,CDATA,TDATA,
C          CORWID,DIST,SMUNIT,SMFILE,MUNIT,MFILE)
C
C      INTEGER      AUNIT,ITR,SMUNIT,RC(30,3),S(10),ROW,COL,MUNIT
C
C      REAL          SDATA(1),CDATA(1),TDATA(1),PL,CORWID,DIST(3,2),TCP(6)
C
C      CHARACTER*40  SKEY(1),DISP(30)
C      CHARACTER*15  STAMP,STAMPS(2)
C      CHARACTER*7   SMFILE,MFILE
C      CHARACTER*1   INP(1),FORMFD
C
C      LOGICAL       AFLAG
C
C      DATA S/ 10*0 /
C
C      S(2)= SMUNIT
C
C      INITIALIZE THE DISPLAY TEXT ARRAY
C
C      DO 1 I=1,30
C          WRITE(DISP(I)(1:40),'(40X)' )
C      1   CONTINUE
C
C      INITIALIZE THE FILE STORAGE SYSTEM
C
C      CALL MENUADR(' ',1,1,0,-1,0,0)
C
C      PROCESS MENU ID=250
C
C      CALL MENUSV(SMFILE,250,RC,30,SMUNIT)
C
C      CALL TIME(STAMP)
C      DISP(1)(1:15)= STAMP(1:15)
C      STAMPS(1)= STAMP
C      CALL DATE(STAMP)
C      DISP(2)(1:15)= STAMP(1:15)
C      STAMPS(2)= STAMP
C      DISP(3)(1:40)= SKEY(1)(1:40)
C      DISP(4)(1:40)= SKEY(2)(1:40)
C      Q= ABS(SDATA(11))
C      WRITE(DISP(5)(1:9),'(F9.2)') Q
C
C      WRITE(DISP(6)(1:5),'(F5.1)') CDATA(1)
C      WRITE(DISP(7)(1:5),'(F5.1)') CDATA(2)
C      WRITE(DISP(8)(1:5),'(F5.1)') CDATA(3)

```

```

C      WRITE(DISP(9)(1:5),'(F5.1)') CDATA(4)
C      DCOR= CDATA(2)+180.
      IF( DCOR.GT.360. ) DCOR= DCOR-360.
      WRITE(DISP(10)(1:5),'(F5.1)') DCOR
      WRITE(DISP(11)(1:5),'(F5.1)') CORWID
C
      IF( CDATA(3) .GT. 0.0 ) THEN
        WRITE(DISP(18)(1:19),'(''OB/DG (SIGMA THETA)''') )
      ENDIF
      DO 20 I=1,3,1
        ID1= I+11
        ID2= I+18
        ID3= I+14
        WRITE(DISP(ID1)(1:8),'(F8.1)') SDATA(I+1)
        IF( CDATA(3) .GT. 0.0 ) THEN
          WRITE(DISP(ID2)(1:11),'(F11.1)') DIST(I,2)
        ENDIF
        WRITE(DISP(ID3)(1:11),'(F11.1)') DIST(I,1)
20    CONTINUE
C
C      IF THE SOURCE STRENGTH WAS CALCULATED VIA THE SPILL EQUATION, SHOW DATA
C
      IF( SDATA(11).LT.0.0 ) THEN
        WRITE(DISP(22)(1:10),'(''SPILL DATA'')')
        WRITE(DISP(23)(1:29),'(''POOL AREA (SQ FT): ''',F9.1)')TDATA(1)
        WRITE(DISP(24)(1:29),'(''POOL TEMP.... (C): ''',F9.1)')TDATA(2)
        WRITE(DISP(25)(1:29),'(''Z FACTOR..... : ''',F9.3)')SDATA(5)
      ENDIF
C
      CALL MENUWR(RC,30,1,25,DISP,0,0,S)
C
      IF THE ARCHIVE MODE IS ACTIVE, WRITE THE DISPLAY TO THE ARCHIVE FILE
      IF (AFLAG) CALL MENUWR(' ',1,1,0,-2,AUNIT,0)
C
      PROCESS THE COMMAND LINE
40    INP(1)=' '
      CALL MENURD(RC,30,30,30,INP,-1)
C
      USER WANTS TO PRINT THE DISPLAY
      IF( INP(1) .EQ. 'P' ) THEN
        OPEN(15,FILE='/DEV/PRT')
        IFORM= 12
        FORMFD= CHAR(IFORM)
        WRITE(15,'(A1)') FORMFD
        CALL MENUWR(' ',1,1,0,-2,15,0)
        WRITE(15,'(A1)') FORMFD
        CLOSE(15)
      ENDIF
C
      USER WANTS TO PLOT THE TOXIC CORRIDOR
      IF( INP(1) .EQ. 'G' ) THEN
        TCP(1)=DIST(1,1)
        TCP(2)=DIST(2,1)
        TCP(3)=DIST(3,1)
        TCP(4)=CORWID
        TCP(5)=DCOR
        CALL TCGRPH(ITR,SMUNIT,SMFILE,MUNIT,MFILE,TCP,
          STAMPS)

```

RETURN
ENDIF

C USER WANTS TO RETURN TO THE CALCULATION MENU
 IF(INP(1) .EQ. 'X') RETURN

GO TO 40
END

*BIGCODE

*SEGMENT *TCGRPH

SUBROUTINE TCGRPH(ITR, SMUNIT, SMFILE, MUNIT, MFILE, TCP, STAMPS)

C*****

C THIS SUBROUTINE PROCESSES THE MAP DATA AND DRAWS THE MAPS ON THE PLOTTER
C VARIABLES PASSED:

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C*****

CHARACTER*48 IO, TT(2)

CHARACTER*40 TEXT(2), TEMF, IOP, FNAME

CHARACTER*20 ANNO

CHARACTER*15 STAMPS(2)

CHARACTER*7 SMFILE, MFILE

CHARACTER*1 CMD(1), INP

INTEGER ITR, SMUNIT, MUNIT, RC(4,3), ST(3), HDR, IS, IE, WCS(3),
* ANGLE, RECNO, SN, ID, NPTS, EFLAG, MARKS(20), SYM

REAL TCP(6), FPI

LOGICAL FLAG, FLAG1, FLAG2

DATA ST /0,0,0/

C 'MARKS' LINKS THE 20 TYPES OF MAP INFORMATION TO HOUSTON PLOTTER SYMBOLS
DATA MARKS/ 0,2,3,0,0,0,0,4,0,0,0,0,0,0,0,5,1,0,0,0/

ITT=-1

FLAG1=.FALSE.

C DISPLAY THE PLOT OPTIONS MENU

20 CALL MENUSV(SMFILE, 301, RC, 4, SMUNIT)

C SCAN THE COMMAND INPUT LINE

25 CMD(1)=' '

CALL MENURD(RC, 4, 1, 1, CMD, ITR)

C CHECK FOR VALID INPUT

IF (INDEX('123X', CMD(1)) .EQ. 0) THEN

CALL MESS(11, RC(2,1), RC(2,2), RC(2,3), 7)

GOTO 25

ENDIF

C THE USER SELECTED TO RETURN

IF (CMD(1) .EQ. 'X') RETURN

C PLOT ONLY CORRIDOR

```

IF (CMD(1) .EQ. '1') THEN
    ST(3)=1
    RC(3,1)=20
    RC(3,2)=1
    RC(3,3)=33
    TT(1)='ENTER SCALE FACTOR (FT/INCH) ==>'
    CALL MENUWR(RC,4,3,3,TT,0,7,ST)
    RC(4,1)=20
    RC(4,2)=35
    RC(4,3)=8
15    TT(1)(1:48)=' '
    CALL MENURD(RC,4,4,4,TT,ITT)
    READ(TT(1)(1:8),'(F8.0)',ERR=15) FPI

C        PLOT THE CORRIDOR USING FPI SCALE FACTOR
    CALL CORPLT(FPI,TCP,STAMPS)
    GOTO 20
ENDIF

C        PLOT CORRIDOR ON OLD MAP
IF (CMD(1) .EQ. '3') THEN
    FLAG1=.TRUE.
    ST(3)=1
    RC(3,1)=20
    RC(3,2)=1
    RC(3,3)=41
    TT(1)='REGISTER MAP AND PRESS RETURN TO CONTINUE'
    CALL MENUWR(RC,4,3,3,TT,0,7,ST)
    READ(ITR,'(A1)') INP
ENDIF

C        PLOT CORRIDOR ON NEW MAP
INP=' '
TEXT(1)(1:40)=' '
TEXT(2)(1:40)=' '

C        DISPLAY THE MENU TO INPUT THE BASE AND MAP FILE NAMES
1000    CALL MENUSV(SMFILE,251,RC,4,SMUNIT)

C        CHECK TO SEE IF ? WAS USED
IF (INP .EQ. '?') THEN
    ST(3)=0
    CALL MENUWR(RC,4,1,2,TEXT,0,1,ST)
    GOTO 150
ENDIF

C        INPUT THE BASE AND MAP FILE NAMES
IST=1
100    CALL MENURD(RC,4,IST,2,TEXT,ITR)

C        SCAN THE COMMAND INPUT LINE
150    CMD(1)=' '
    CALL MENURD(RC,4,3,3,CMD,ITT)

C        THE USER WISHES TO GO BACK TO THE FIRST BASE MAP FILE NAME
IF (CMD(1) .EQ. ' ') THEN
    ST(3)=1
    CALL MESS(4,RC(4,1),RC(4,2),RC(4,3),1)
    CALL MENUWR(RC,4,3,3,CMD,0,1,ST)
    IST=1

```

GOTO 100
ENDIF

```
C      THE USER SELECTED TO RETURN TO THE CORRIDOR OPTIONS MENU
      IF (CMD(1) .EQ. 'X') GOTO 20

C      INVALID INPUT
      IF (CMD(1) .NE. 'C') GOTO 150

C      IF THE USER SPECIFIED A ? FOR BASE MAP DISPLAY LIST
      IF (TEXT(1) .EQ. '?') THEN

C          FIRST CHECK TO SEE IF THE SUPER DIRECTORY IS AVAILABLE
          INQUIRE (FILE=MFILE,EXIST=FLAG)

C          IF NOT AVAILABLE THEN DISPLAY AN ERROR MESSAGE NOT FOUND
          IF (.NOT. FLAG) THEN
              ST(3)=1
              CMD(1)=' '
              CALL MENUWR(RC,4,3,3,CMD,0,1,ST)
              CALL MESS(5,RC(4,1),RC(4,2),RC(4,3),7)
              IST=1
              GOTO 100
          ENDIF

C          CALL ROUTINE TO DISPLAY THE LIST OF BASE MAPS AVAILABLE
          FNAME=MFILE
          CALL MQST(TEXT(1),MUNIT,FNAME,SMUNIT,SMFILE,ITR,EFLAG,
*              42,252)

C          CHECK THAT NO ERROR OCCURRED WHILE TRYING TO DISPLAY THE LIST.
C          I.E. THERE ARE NO ENTRIES IN THE SUPER DIRECTORY
          IF (EFLAG .NE. 0) THEN
              ST(3)=1
              CMD(1)=' '
              CALL MENUWR(RC,4,3,3,CMD,0,1,ST)
              CALL MESS(5,RC(4,1),RC(4,2),RC(4,3),7)
              IST=1
              GOTO 100
          ENDIF

          INP='*'

C          IF MAP FILE NAME IS KNOWN, THEN GO BACK AND FILL IN THE BASE MAP
C          FILE NAME IN THE MENU AND CONTINUE
          IF (TEXT(2) .NE. '?') GOTO 1000
          ENDIF

C          CHECK IF THE BASE MAP EXISTS THAT THE USER SPECIFIED (PACKING REQUIRED)
          TEMP=TEXT(1)
          CALL PACK(TEMP,J)
          INQUIRE (FILE=TEMP,EXIST=FLAG)

C          IF BASE MAP FILE DOES NOT EXIST, THEN DISPLAY ERROR MESSAGE
          IF (.NOT. FLAG) THEN
              ST(3)=1
              CMD(1)=' '
              CALL MENUWR(RC,4,3,3,CMD,0,1,ST)
              IST=1
              IF (INP .EQ. '*')
*                  CALL MENUSV(SMFILE,251,RC,4,SMUNIT)
```

```

        CALL MESS(5, RC(4, 1), RC(4, 2), RC(4, 3), 7)
        GOTO 100
    ENDIF

C      CHECK FOR A ? FOR THE MAP FILE NAME
      IF (TEXT(2) .EQ. '?') THEN

C          CALL ROUTINE TO DISPLAY A LIST OF THE MAP FILES CONTAINED IN THE
C          SELECTED BASE MAP
          CALL MQST(TEXT(2), MUNIT, TEMP, SMUNIT, SMFILE, ITR, EFLAG,
*              50, 253)

C          CHECK THAT NO ERRORS OCCURRED WHILE DISPLAYING THE LIST. I.E.
C          THAT NO ENTRIES WERE FOUND IN THE BASE MAP FILE
          IF (EFLAG .NE. 0) THEN
              ST(3)=1
              CMD(1)=' '
              CALL MENUWR(RC, 4, 3, 3, CMD, 0, 1, ST)
              IST=2
              IF (INP .EQ. '*')
*                  CALL MENUSV(SMFILE, 251, RC, 4, SMUNIT)
                  CALL MESS(5, RC(4, 1), RC(4, 2), RC(4, 3), 7)
                  GOTO 100
              ENDIF

              INP='*'
              GOTO 1000

                  ENDIF

C      CHECK THAT THE BASE MAP .MM FILE EXISTS
      IF ((J+1) .GT. 38) THEN
          TEMP(38:40)='.MM'
      ELSE
          TEMP(J+1:J+3)='.MM'
      ENDIF
      INQUIRE (FILE=TEMP, EXIST=FLAG)
      IF (.NOT. FLAG) THEN
          ST(3)=1
          CMD(1)=' '
          CALL MENUWR(RC, 4, 3, 3, CMD, 0, 1, ST)
          IST=1
          IF (INP .EQ. '*') CALL MENUSV(SMFILE, 251, RC, 4, SMUNIT)
          CALL MESS(5, RC(4, 1), RC(4, 2), RC(4, 3), 7)
          GOTO 100
      ENDIF

C      OPEN THE BASE FILE THE USER SELECTED TO CHECK AND SEE IF THE MAP FILE
C      THE USER SELECTED CAN BE FOUND
      TEMP=TEXT(1)
      CALL PACK(TEMP, J)
      OPEN (MUNIT, FILE=TEMP, STATUS='OLD', ACCESS='DIRECT', RECL=50,
*          FORM='FORMATTED')

C      READ IN THE HEADER DATA TELLING HOW MANY RECORDS ARE IN THE FILE
      READ(MUNIT, '(I4)', REC=1) HDR

C      SEARCH THE BASE MAP FILE FOR A MATCH TO THE SELECTED FILE NAME
      DO 200 I=2, HDR, 1
          READ(MUNIT, '(A48)', REC=I) IO
          IF (IO(1:40) .EQ. TEXT(2)) GOTO 250
200    CONTINUE

```

```

C      NO MATCH FOUND, DISPLAY AN ERROR MESSAGE
      ST(3)=1
      CMD(1)=' '
      CALL MENUWR(RC,4,3,3,CMD,0,1,ST)
      CALL MESS(5,RC(4,1),RC(4,2),RC(4,3),7)
      IST=2
      GOTO 100

C      MATCH FOUND IN THE BASE MAP FILE
250    CALL MESS(4,RC(4,1),RC(4,2),RC(4,3),1)
      CLOSE (MUNIT)

C      GET THE STARTING AND STOPPING LOCATION IN THE BASE MAP .MM FILE
C      WHERE THE SELECTED MAP DATA CAN BE FOUND
      READ(IO(41:48),'(2I4)') IS,IE
      ISS=IS
      IEE=IE

C      SET THE BASE MAP .MM FILE NAME
      IF ((J+1) .GT. 38) THEN
          TEMP(38:40)='.MM'
      ELSE
          TEMP(J+1:J+3)='.MM'
      ENDIF

C      OPEN THE BASE FILE .MM TO PROCESS THE MAP
      OPEN (MUNIT,FILE=TEMP,STATUS='OLD',ACCESS='DIRECT',RECL=42,
*        FORM='FORMATTED')

C      READ IN THE FIRST RECORD, WHICH SHOULD BE THE MAP NAME
      READ(MUNIT,'(A40)',REC=IS) IOP

C      THE MAP NAME IN THE FILE MUST AGREE WITH THE MAP NAME GIVEN
      IF (IOP .NE. TEXT(2)) THEN
          ST(3)=1
          CMD(1)=' '
          CALL MENUWR(RC,4,3,3,CMD,0,1,ST)
          CALL MESS(20,RC(4,1),RC(4,2),RC(4,3),7)
          CLOSE (MUNIT)
          GOTO 150
      ENDIF

C      READ IN THE VIRTUAL LOWER AND UPPER X AND Y
      READ(MUNIT,'(2F10.2)',REC=IS+7) VLX,VLY
      READ(MUNIT,'(2F10.2)',REC=IS+8) VUX,VUY

C      INITIALIZE THE PLOT DEVICE, DRAW A FRAME AROUND THE MAP AREA AND TITLE
C      THE PLOT, CONVERT THE FLOATING POINT DATA TO INTEGER DATA
      IVLX=VLX
      IVLY=VLY
      IVUX=VUX
      IVUY=VUY
      CALL PLINIT(IVLX,IVLY,IVUX,IVUY)

C      IF OLD MAP THEN SELECT THE SITE WHERE THE CORRIDOR IS TO GO AND PLACE
C      THE LEGEND ON THE MAP WITH THE CORRIDOR INFORMATION
      IF (FLAG1) THEN
          CALL TCLIST(ITR,SMUNIT,SMFILE,MUNIT,ISS,IEE,TCP,FLAG2)
          CALL LEGEND(IVUX,IVUY,TCP,STAMPS)

```

```

        CALL PLDONE
        CLOSE (MUNIT)
        FLAG1=.FALSE.
        GOTO 20
    ENDIF

C      FOR A NEW MAP DRAW A FRAME AROUND IT
    CALL FRAME(TEXT(2), IVLX, IVLY, IVUX, IVUY, 1, 0)

C      SET THE RECORD POINTER TO THE FIRST DATA RECORD
    IS=IS+15

C      CHECK TO SEE IF THE END OF THIS MAP FILE HAS BEEN REACHED
300    IF (IS .GT. IE) THEN

C          FLAG2 ENABLES THE USER TO DRAW ONLY THE MAP WITHOUT
C          DISPLAYING ANY CORRIDORS ON IT.
C          TRUE ==> DISPLAY THE CORRIDOR DATA
C          FALSE ==> DO NOT DISPLAY CORRIDOR DATA
        FLAG2=.TRUE.
        CALL TCLIST(ITR, SMUNIT, SMFILE, MUNIT, ISS, IEE, TCP, FLAG2)
        IF (FLAG2) CALL LEGEND(IVUX, IVUY, TCP, STAMPS)
        CALL PLDONE
        CLOSE (MUNIT)
        GOTO 20
    ENDIF

C      READ IN THE DATA ID RECORD
C      WCS IS AN INTEGER ARRAY THAT HOLDS THE W(EIGHT)C(OLOR)S(TYLE) OF
C      THE ITEM TO BE DRAWN
    READ(MUNIT, '(6I2,F4.2,2I4,A16)', REC=IS) ID, SN, NPTS, WCS(1), WCS(2),
*      WCS(3), SCALE, ANGLE, RECNO, ANNO

C      READ IN THE ANNOTATION FOR DATA ITEM
    READ(MUNIT, '(2F10.2,A20)', REC=IS+1) AX, AY, ANNO

C      IF THE ANNOTATION IS NOT BLANK THEN PLOT THE TEXT
    IF (ANNO .NE. ' ') THEN
        IAX=AX
        IAY=AY
        CALL PLTEXT(IAX, IAY, 2, 0, ANNO, 20, WCS)
    ENDIF

C      ID IS A POINT, THIS MEANS A MARKER NEEDS TO BE DRAWN
    IF (ID .EQ. 10) THEN
        READ(MUNIT, '(2F10.2)', REC=IS+2) VX, VY
        IVX=VX
        IVY=VY
        SYM= MARKS(SN)
        CALL MARKER(SYM, IVX, IVY, WCS, SCALE, ANGLE)
        IS=IS+3
        GOTO 300
    ENDIF

C      ID IS A LINE
    IF (ID .EQ. 11) THEN
        IS=IS+3
        NREC=NPTS/2+MOD(NPTS, 2)
        IREC=1
        IPTS=0
    
```

AD-A141 405

A REAL-TIME AIR DISPERSION MODELING SYSTEM(U) SIERRA
GEOPHYSICS INC REDMOND WA D E BLEEKER ET AL. APR 84
AFESC/ESL-TR-83-63 F08635-82-C-0374

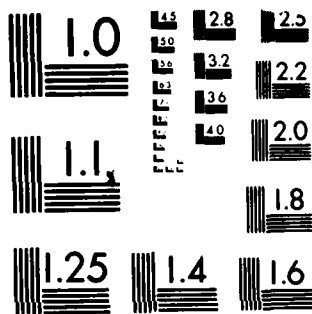
2/2

UNCLASSIFIED

F/G 9/2

NL

END
DATE
FORM D
7-84
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

```

C      ARE ANY MORE LINE RECORDS TO BE READ
400    IF (IREC .GT. NREC) THEN
          IS=IS+NREC
          GOTO 300
        ENDIF

C      FIND OUT HOW MANY POINTS ARE VALID ON THE RECORD
      IF ((IPTS+2) .GT. NPTS) THEN
          IFLG=1
        ELSE
          IFLG=2
        ENDIF

      IPTS=IPTS+IFLG

C      READ IN THE RECORD
      READ(MUNIT,'(4F10.2)',REC=IS+IREC-1) VX,VY,VX1,VY1

      IVX=VX
      IVY=VY
      IF (IREC .EQ. 1) THEN
          CALL MOVETO(IVX,IVY)
        ELSE
          CALL LINETO(IVX,IVY,WCS)
        ENDIF

C      IF THERE ARE TWO POINTS DISPLAY THE SECOND
      IVX=VX1
      IVY=VY1
      IF (IFLG .EQ. 2) CALL LINETO(IVX,IVY,WCS)
      IREC=IREC+1
      GOTO 400
    ENDIF

C      ID IS A POLYGON -- END POINTS MUST BE CONNECTED
      IF (ID .EQ. 12) THEN
          IS=IS+3
          NREC=NPTS/2+MOD(NPTS,2)
          IREC=1
          IPTS=0

C      ARE ANY MORE LINE RECORDS TO BE READ
500    IF (IREC .GT. NREC) THEN
          CALL LINETO(ISX,ISY,WCS)
          IS=IS+NREC
          GOTO 300
        ENDIF

C      FIND OUT HOW MANY POINTS ARE VALID ON THE RECORD
      IF ((IPTS+2) .GT. NPTS) THEN
          IFLG=1
        ELSE
          IFLG=2
        ENDIF

      IPTS=IPTS+IFLG

C      READ IN THE RECORD
      READ(MUNIT,'(4F10.2)',REC=IS+IREC-1) VX,VY,VX1,VY1

      IVX=VX

```

```

IVY=VY
IF (IREC .EQ. 1) THEN
    CALL MOVETO(IVX, IVY)
    ISX=IVX
    ISY=IVY
ELSE
    CALL LINETO(IVX, IVY, WCS)
ENDIF

```

```

C      IF THERE ARE TWO POINTS DISPLAY THE SECOND
      IVX=VX1
      IVY=VY1
      IF (IFLG .EQ. 2) CALL LINETO(IVX, IVY, WCS)
      IREC=IREC+1
      GOTO 500
      ENDIF

```

```

END
SUBROUTINE MQST(TEXT, MUNIT, FNAME, SMUNIT, SMFILE, ITR, EFLAG,
*          IREC, IMN)

```

```

C*****

```

```

C  THIS SUBROUTINE DISPLAYS A LIST OF THE BASE FILE NAMES OR MAP FILE NAMES
C  VARIABLES PASSED:

```

```

C      MUNIT  -  UNIT # TO OPEN THE SUPER DIRECTORY OR BASE MAP FILES ON
C      FNAME  -  NAME OF FILE TO BE OPENED ON MUNIT
C      SMUNIT -  UNIT # TO OPEN MENU FILE ON (SMFILE)
C      SMFILE -  MENU FILE NAME
C      ITR    -  INTERACTIVE TERMINAL READ UNIT
C      IREC   -  RECORD LENGTH TO OPEN THE FILE WITH
C              42 ==> SUPER DIRECTORY FILE
C              50 ==> BASE MAP FILE
C      IMN    -  MENU NUMBER TO DISPLAY
C              252 ==> SUPER DIRECTORY MENU
C              251 ==> BASE MAP MENU

```

```

C      VARIABLES RETURNED:
C      TEXT   -  NAME OF BASE OR MAP FILE USER SELECTED
C      EFLAG  -  FLAG TO DETERMINE IF THERE WERE ANY ENTRIES IN THE FILE
C              1 ==> NO ENTRIES FOUND
C              0 ==> ENTRIES WERE FOUND

```

```

C*****

```

```

CHARACTER*70 OUT(19)
CHARACTER*40 IO, TEXT, FNAME
CHARACTER*7  SMFILE
CHARACTER*3  CMD(1)

```

```

INTEGER      SMUNIT, RC(19, 3), ST(3), ICNT, ITR, MUNIT, EFLAG, HD,
*            IREC, IMN

```

```

LOGICAL      FLAG1

```

```

DATA ST/0, 0, 0/

```

```

C      OPEN THE FILE AND READ IN THE HEADER RECORD TO FIND OUT HOW MANY RECORDS
C      ARE IN THE FILE
      OPEN (MUNIT, FILE=FNAME, STATUS='OLD', RECL=IREC, ACCESS='DIRECT',
*          FORM='FORMATTED')

```

```

READ(MUNIT,'(I4)',REC=1) HD

C   SET THE ERROR FLAG
    EFLAG=0
    IF (HD .LE. 1) THEN
        EFLAG=1
        RETURN
    ENDIF

5   ICNT=2
    FLAG1=.FALSE.

C   FORM THE LIST OF FILES THE USER CAN SELECT FROM
    DO 20 I=2,HD,1
        IF (.NOT. FLAG1) FLAG1=.TRUE.
        READ(MUNIT,'(A40)',REC=I) IO
        OUT(ICNT)(1:70)=' '
        WRITE(OUT(ICNT)(1:3),'(I3)') I
        OUT(ICNT)(6:70)=IO

C   CAN ONLY DISPLAY 19 NAMES ON THE SCREEN AT ONCE
    IF (ICNT .EQ. 19) THEN
        CALL MENUSV(SMFILE,IMN,RC,19,SMUNIT)
        CALL MENUWR(RC,19,2,19,OUT,0,1,ST)

C   INPUT THE USERS CHOICE
15  CMD(1)=' '
    CALL MENURD(RC,19,1,1,CMD,ITR)

C   USER SELECTED TO RETURN, CLOSE THE MAP UNIT FIRST
    IF (CMD(1) .EQ. 'X ') THEN
        CLOSE (MUNIT)
        RETURN
    ENDIF

C   THE USER WANTS TO CONTINUE
    IF (CMD(1) .EQ. 'C ') THEN
        ICNT=2
        FLAG1=.FALSE.
        GOTO 20
    ENDIF

C   CHECK FILE THE USER SELECTED IS IN THE USABLE RANGE
    READ(CMD(1)(1:3),'(I3)',ERR=15) II
    IF ((II .LT. 2).OR.(II .GT. HD)) GOTO 15

C   USER SELECTED A USABLE FILE, STORE IT IN TEXT, CLOSE THE FILE, AND
C   RETURN
    READ(MUNIT,'(A40)',REC=II) TEXT
    CLOSE (MUNIT)
    RETURN
                                ENDIF

    ICNT=ICNT+1
20  CONTINUE

C   THERE WERE LESS THAN 19 FILES TO DISPLAY
    IF (.NOT. FLAG1) GOTO 5

C   DISPLAY THE MENU AND THE FILE NAMES
    CALL MENUSV(SMFILE,IMN,RC,19,SMUNIT)

```

```

      CALL MENUWR(RC,19,2,ICNT-1,OUT,0,1,ST)

C      INPUT THE USERS CHOICE
25     CMD(1)=' '
      CALL MENURD(RC,19,1,1,CMD,ITR)

C      THE USER WANTS TO RETURN
      IF (CMD(1) .EQ. 'X ') THEN
          CLOSE (MUNIT)
          RETURN
      ENDIF

C      THE USER WANTS TO CONTINUE VIEWING THE LIST
      IF (CMD(1) .EQ. 'C ') THEN
          FLAG1=.FALSE.
          GOTO 5
      ENDIF

C      CHECK THAT THE USERS INPUT IS VALID
      READ(CMD(1)(1:3),'(I3)',ERR=25) II
      IF ((II .LT. 2).OR.(II .GT. HD)) GOTO 25

C      THE INPUT IS VALID, STORE THE VALID FILE NAME IN TEXT
      READ(MUNIT,'(A40)',REC=II) TEXT
      CLOSE (MUNIT)
      RETURN

END
SUBROUTINE TCLIST(ITR,SMUNIT,SMFILE,MUNIT,IS,IE,TCP,FLAG)

C*****
C  THIS SUBROUTINE IS USED TO INPUT THE SITE (COORDINATES) WHERE THE TOXIC
C  CORRIDOR IS TO BE PLOTTED
C  VARIABLES PASSED:
C
C      ITR      -  INTERACTIVE TERMINAL READ UNIT
C      SMUNIT   -  UNIT # TO OPEN MENU FILE (SMFILE) ON
C      SMFILE   -  MENU FILE NAME
C      MUNIT    -  UNIT # TO OPEN THE BASE MAP .MM FILE ON
C      IS       -  STARTING RECORD OF INTEREST IN BASE MAP .MM FILE
C      IE       -  ENDING RECORD OF INTEREST IN BASE MAP .MM FILE
C      TCP      -  TOXIC CORRIDOR PARAMETERS NEEDED TO PLOT THE CORRIDOR
C
C  VARIABLES RETURNED:
C
C      FLAG     -  FLAG TO DETERMINE WHETHER THE LEGEND WILL BE DRAWN ON THE MAP
C                  TRUE ==> THE LEGEND IS TO BE DRAWN
C                  FALSE ==> THE LEGEND IS NOT TO BE DRAWN
C*****

      CHARACTER*20  TEXT(1)
      CHARACTER*10  DIG(2)
      CHARACTER*7   SMFILE
      CHARACTER*1   CMD(1),INP

      INTEGER       ITR,SMUNIT,IX,IY,IF1,RC(5,3),ITT,ST(3),WCS(3),NCOR

      REAL          TCP(6),DC(4),CORDIR,CORWID

      LOGICAL       FLAG

```

```

DATA ST /0,0,0/
DATA WCS /1,1,1/

ITT=-1
INP=' '
TEXT(1)(1:20)=' '
DIG(1)(1:10)=' '
DIG(2)(1:10)=' '

C   DISPLAY THE MENU
1000 CALL MENUSV(SMFILE,254,RC,5,SMUNIT)

C   CHECK THAT A ? WAS USED FOR THE SITE
    IF (INP .EQ. '?') THEN
        ST(3)=1
        CALL MENUWR(RC,5,1,1,TEXT,0,1,ST)
        WRITE(DIG(1)(1:10),'(I7)') IX
        WRITE(DIG(2)(1:10),'(I7)') IY
        CALL MENUWR(RC,5,2,3,DIG,0,1,ST)
        GOTO 100
    ENDIF

C   READ IN THE SITE NAME
25   CALL MENURD(RC,5,1,1,TEXT,ITR)

C   IF A ? WAS ENTERED FOR THE SITE, THEN DISPALY A LIST OF ALL POSSIBLE
C   SOURCES
    IF (TEXT(1) .EQ. '?') THEN
        CALL TCSITE(ITR,SMUNIT,SMFILE,MUNIT,IS,IE,TEXT(1),IX,IY)
        INP='*'
        GOTO 1000
    ENDIF

C   READ IN THE COORDINATES
    IST=2
30   CALL MENURD(RC,5,IST,3,DIG,ITT)
    READ(DIG(1)(1:10),'(F10.0)',ERR=50) VX
    READ(DIG(2)(1:10),'(F10.0)',ERR=55) VY
    IX=VX
    IY=VY
    GOTO 100

C   ERROR RECOVERY FOR THE COORDINATES
50   IST=2
    CALL MESS(1,RC(5,1),RC(5,2),RC(5,3),7)
    GOTO 30
55   IST=3
    CALL MESS(1,RC(5,1),RC(5,2),RC(5,3),7)
    GOTO 30

C   SCAN THE COMMAND INPUT LINE
100  CMD(1)=' '
    CALL MENURD(RC,5,4,4,CMD,ITT)

C   USER WANTS TO GO BACK TO THE SITE ENTRY
    IF (CMD(1) .EQ. ' ') THEN
        ST(3)=1
        CALL MENUWR(RC,5,4,4,CMD,0,1,ST)
        CALL MESS(4,RC(5,1),RC(5,2),RC(5,3),1)
    
```

```

                                GOTO 25
                                ENDIF

C   USER WANTS TO RETURN, AND NOT DRAW THE LEGEND
    IF (CMD(1) .EQ. 'X') THEN
        FLAG=.FALSE.
        RETURN
    ENDIF

C   USER WANTS TO DRAW THE CORRIDOR ON THE MAP AND A LEGEND
    IF (CMD(1) .EQ. 'P') THEN
        PLOT THE CIRCLES FOR 10, 30, 60 PEL AND 1800 FT
        PRIORITY ZONE

        DC(1)= TCP(1)
        DC(2)= TCP(2)
        DC(3)= TCP(3)
        DC(4)= 1800.0
        CORWID= TCP(4)
        CORDIR= TCP(5)

        CALL DRWCOR(IX,IY,DC,4,CORWID,CORDIR)

    RETURN
    ENDIF

GOTO 100
END
SUBROUTINE TCSITE(ITR,SMUNIT,SMFILE,MUNIT,IS,IE,SITE,IX,IY)

C*****
C   THIS SUBROUTINE SEARCHES THE BASE MAP .MM FILE AND LISTS ALL THE POSSIBLE
C   SOURCES THAT COULD BE USED TO DRAW THE CORRIDOR AROUND
C   VARIABLES PASSED:
C
C   ITR      -  INTERACTIVE TERMINAL READ UNIT
C   SMUNIT   -  UNIT # TO OPEN MENU FILE (SMFILE) ON
C   SMFILE   -  MENU FILE NAME
C   MUNIT    -  UNIT # TO OPEN BASE MAP .MM ON
C   IS       -  STARTING RECORD IN BASE MAP .MM
C   IE       -  ENDING RECORD IN BASE MAP .MM
C
C   VARIABLES RETURNED:
C
C   SITE     -  SITE THAT THE USER SELECTED
C   IX       -  X COORDINATE OF THE SITE
C   IY       -  Y COORDINATE OF THE SITE
C*****

CHARACTER*70  OUT(19)
CHARACTER*40  ID
CHARACTER*20  SITE
CHARACTER*7   SMFILE
CHARACTER*3   CMD(1)

INTEGER      SMUNIT, RC(20,3), ST(3), ICNT, ITR, MUNIT, EFLAG, HD,
*            IREC, IS, IE, IX, IY

REAL         CO(19,2)

```

LOGICAL FLAG1

DATA ST/0,0,0/

```
C    INITIALIZE THE CO-ORDINATE ARRAY TO ALL ZEROS
DO 1 I=1,19,1
    CO(I,1)=0.
    CO(I,2)=0.
1    CONTINUE
    EFLAG=1

5    ICNT=2

C    IREC POINTS TO THE FIRST RECORD OF MAP DATA
    IREC=IS+15
    FLAG1=.FALSE.
20   IF (.NOT. FLAG1) FLAG1=.TRUE.
    READ(MUNIT,'(I2,2X,I2,18X,I2)',REC=IREC) NSYM,NPTS,LINK
    READ(MUNIT,'(A40)',REC=IREC+1) IO
    IREC=IREC+2
    IF (NSYM.NE. 10) IREC=IREC+1

C    INPUT THE X AND Y COORDINATE OF THE SITE
    READ(MUNIT,'(2F10.2)',REC=IREC) CO(ICNT,1),CO(ICNT,2)

C    SET IREC TO POINT TO THE NEXT RECORD
    IREC=IREC+(NPTS/2)+MOD(NPTS,2)

C    IF LINK = 1 THEN THE ITEM IN THE FILE IS A POTENTIAL SOURCE
    IF (LINK.NE. 1) GOTO 21

C    EFLAG IS USED TO DETERMINE WHETHER THERE WERE ANY ITEMS IN THE FILE
C    THAT WERE POTENTIAL SOURCES
C    EFLAG =0 ==> NO POTENTIAL SOURCES
C    EFLAG =1 ==> POTENTIAL SOURCES
    EFLAG=0

C    STORE OF THE SITE IN AN ARRAY
    OUT(ICNT)(1:70)=' '
    WRITE(OUT(ICNT)(1:3),'(I3)') ICNT
    OUT(ICNT)(6:25)=IO(21:40)

C    CAN ONLY DISPLAY 19 SITES ON THE SCREEN AT A TIME
    IF (ICNT.EQ. 19) THEN

C        DISPLAY THE MAIN MENU AND THE LIST OF SITES
        CALL MENUSV(SMFILE,255,RC,20,SMUNIT)
        CALL MENUWR(RC,20,2,19,OUT,0,1,ST)

C        INPUT THE USERS COMMAND
15        CMD(1)=' '
        CALL MENURD(RC,20,1,1,CMD,ITR)

C        USER SELECTED TO RETURN
        IF (CMD(1).EQ. 'X ') RETURN

C        USER WANTS TO CONTINUE AND SEE MORE OF LIST
        IF (CMD(1).EQ. 'C ') THEN
            ICNT=2
            FLAG1=.FALSE.
```

```

                                GOTO 20
                                ENDIF

G      CHECK THAT THE USER SELECTED A VALID SITE
      READ(CMD(1)(1:3),'(I3)',ERR=15) II
      IF ((II .LT. 2).OR.(II .GT. 19)) GOTO 15

C      STORE OFF THE SITE AND COORDINATES AND RETURN
      SITE(1:20)=OUT(II)(6:25)
      IX=CO(II,1)
      IY=CO(II,2)
      RETURN
                                ENDIF

      ICNT=ICNT+1
21     IF (IREC .LE. IE) GOTO 20
      IF (.NOT. FLAG1) GOTO 5

C      DISPLAY THE MENU AND ANY SITES NAMES FOUND
      CALL MENUSV(SMFILE,255,RC,20,SMUNIT)

C      IF THERE ARE NO SITES WHICH ARE POTENTIAL SOURCES DISPLAY A MESSAGE
C      AND ALLOW THE USER TO ONLY RETURN
      IF (EFLAG .NE. 0) THEN
                                CALL MESS(5,RC(20,1),RC(20,2),RC(20,3),7)
24     CMD(1)=' '
                                CALL MENURD(RC,20,1,1,CMD,ITR)
                                IF (CMD(1) .NE. 'X ') GOTO 24
                                RETURN
                                ENDIF
      CALL MENUWR(RC,20,2,ICNT-1,OUT,0,1,ST)

C      INPUT THE USERS SELECTION
25     CMD(1)=' '
      CALL MENURD(RC,20,1,1,CMD,ITR)

C      USER SELECTED TO RETURN
      IF (CMD(1) .EQ. 'X ') RETURN

C      USER SELECTED TO CONTINUE VIEWING THE LIST
      IF (CMD(1) .EQ. 'C ') THEN
                                FLAG1=.FALSE.
                                GOTO 5
                                ENDIF

C      CHECK THAT THE USER SELECTED A VALID SITE
      READ(CMD(1)(1:3),'(I3)',ERR=25) II
      IF ((II .LT. 2).OR.(II .GT. 19)) GOTO 25

C      STORE OFF THE SITE AND COORDINATES
      SITE(1:20)=OUT(II)(6:25)
      IX=CO(II,1)
      IY=CO(II,2)

      RETURN
      END
      SUBROUTINE DRWCOR(IX,IY,TCL,NCOR,CORWID,CORDIR)

C
C      ARGUMENT      TYPE      DESCRIPTION
C      IX            INTEGER*4  VIRTUAL X COORDINATE OF CORRIDOR STARTING POINT
C      IY            "          "          Y          "          "          "          "

```

```

C      TCL(NCOR)  REAL*4      TCL(1):  10 MIN PEL DISTANCE
C      TCL(2):  30  "  "  "
C      TCL(3):  60  "  "  "
C      TCL(4-NCOR):  OPTIONAL CORRIDOR LENGTHS
C      NCOR      INTEGER*4    NUMBER OF CORRIDORS TO PLOT
C      CORWID    REAL*4      CORRIDOR WITH (DEGREES)
C      CORDIR    REAL*4      "      DIRECTION (DEGREES)
C
C      INTEGER    IX, IY, NCOR
C      REAL      TCL(NCOR), CORWID, CORDIR
C
C      PROGRAM VARIABLES
C
C      INTEGER    WCS(3), IR, IX1, IY1, IX2, IY2, ICORL
C      REAL      ANG1, ANG2, RADCOR, RADDIR
C      CHARACTER*18 COORD(1)
C
C
C      WCS(1)= 0
C      WCS(3)= 1
C      ICORL= -1.
C      DO 10 I=1,3,1
C        IF( TCL(I).GT.0.) THEN
C          IR=TCL(I)
C          ICORL= MAX0(IR, ICORL)
C          WCS(2)=I+4
C          CALL CIRCLE(IX, IY, IR, WCS)
C        ENDIF
C      CONTINUE
10
C
C      IF( NCOR.GT.3 ) THEN
C        WCS(3)= 0
C        DO 20 I=4, NCOR, 1
C          IF( TCL(I).GT.0.) THEN
C            IR=TCL(I)
C            ICORL= MAX0(IR, ICORL)
C            WCS(2)=I+4
C            CALL CIRCLE(IX, IY, IR, WCS)
C          ENDIF
C        CONTINUE
20
C      ENDIF
C
C      IF( ICORL.GT.0 ) THEN
C        RADCOR= CORWID/57.3
C        RADDIR= (90.-CORDIR)/57.3
C        ANG1= RADDIR - (RADCOR/2.0)
C        ANG2= RADDIR + (RADCOR/2.0)
C        IX1= IX+( ICORL*COS(ANG1) )
C        IY1= IY+( ICORL*SIN(ANG1) )
C        IX2= IX+( ICORL*COS(ANG2) )
C        IY2= IY+( ICORL*SIN(ANG2) )
C        CALL MOVETO(IX, IY)
C        WCS(1)= 0
C        WCS(2)= 1
C        WCS(3)= 0
C        CALL LINETO(IX1, IY1, WCS)
C        CALL MOVETO(IX, IY)
C        CALL LINETO(IX2, IY2, WCS)
C      ENDIF

```

C

RETURN
END

SUBROUTINE LEGEND(VUX,VUY,TCP,STAMPS)

C

C

LEGEND PLOTS TIME, DATE AND CORRIDOR INFORMATION IN TEXT FORM

C

C

ARGUMENT	TYPE	DESCRIPTION
VUX	INTEGER	UPPER LEFT X VIRTUAL COORDINATE
VUY	INTEGER	" " Y " "
TCP(5)	INTEGER	CORRIDOR LENGTHS, WIDTH AND DIRECTION
TITLE(1)	CHARACTER*40	MAP TITLE
STAMPS(2)	CHARACTER*15	TIME AND DATE STAMPS

C

C

INTEGER	VUX,VUY
REAL	TCP(5)
CHARACTER*15	STAMPS(2)

C

C

C

PROGRAM VARIABLES

C

INTEGER	SPEL(3),WCS(3),SIZE,IX,IY,IX1,IY1
CHARACTER*80	TS(1)

C

C

C

INITIALIZE LINE EXPOSURE LIMIT (SPEL) AND LINE ATTRIBUTE (WCS) ARRAYS

C

SPEL(1)= 10
SPEL(2)= 30
SPEL(3)= 60
WCS(1)= 0
WCS(2)= 1
WCS(3)= 0

C

CALL CHRSLZ(3,SIZE)
SIZE= 1.5*SIZE
IX= VUX+(4*SIZE)
IY= VUY-(4*SIZE)
WRITE(TS(1)(1:22),'('TIME: ',A15)') STAMPS(1)
CALL PLTEXT(IX,IY,3,0,TS(1),22,WCS)
IY= IY-SIZE
WRITE(TS(1)(1:22),'('DATE: ',A15)') STAMPS(2)
CALL PLTEXT(IX,IY,3,0,TS(1),22,WCS)
IY= IY-(2*SIZE)
CALL PLTEXT(IX,IY,3,0,'CORRIDOR INFORMATION',26,WCS)
IY= IY-(1.2*SIZE)
CALL PLTEXT(IX,IY,3,0,'OCEAN BREEZE - DRY GULCH EQUATION',33,WCS)
IY= IY-SIZE
CALL PLTEXT(IX,IY,3,0,'(BASED ONLY ON DELTA T)',26,WCS)
WRITE(TS(1)(1:18),'('DIRECTION: ',F6.1)') TCP(5)
IY= IY-(1.25*SIZE)
CALL PLTEXT(IX,IY,3,0,TS(1),18,WCS)
WRITE(TS(1)(1:18),'('WIDTH : ',F6.1)') TCP(4)
IY= IY-SIZE
CALL PLTEXT(IX,IY,3,0,TS(1),18,WCS)

C

C

PLOT CORRIDOR LINES AND DESCRIPTIONS

C

```

      IY= IY-(2*SIZE)
      IX1= IX+(5*SIZE)
      DO 10 I=1, 3
        IF( TCP(I).GT.0.0 ) THEN
          WCS(2)= I+4
          WCS(3)= 1
          CALL MOVETO(IX,IY)
          CALL LINETO(IX1,IY,WCS)
          WCS(2)= 1
          WCS(3)= 0
          WRITE(TS(1)(1:16),'('' : '',I2,''' MIN SPEL''')'') SPEL(I)
          IY1= IY- (SIZE/3)
          CALL PLTEXT(IX1,IY1,3,0,TS(1),16,WCS)
          IY= IY-SIZE
        ENDIF
10    CONTINUE
      WCS(2)= 8
      WCS(3)= 0
      CALL MOVETO(IX,IY)
      CALL LINETO(IX1,IY,WCS)
      WCS(2)= 1
      IY1= IY-(SIZE/3)
      CALL PLTEXT(IX1,IY1,3,0,' : PRIORITY ZONE',18,WCS)
      IY1= IY1-SIZE
      CALL PLTEXT(IX1,IY1,3,0,' (1800 FEET) ',18,WCS)
C
      RETURN
      END

      SUBROUTINE SCLPLT(IX,IY,FPI)
C
C      ARGUMENT          TYPE          DESCRIPTION
C      IX                INTEGER*4      VIRTUAL LOWER LEFT X COORDINATE OF SCALE PLOT
C      IY                INTEGER*4      " " " Y " " " "
C      FPI               REAL*4         MAP SCALE (FT/INCH)
C
C      PROGRAM VARIABLES
C      INTEGER          IXX,IYY,IX1,IY1,IY2,WCS(3),IFPI
C      REAL              VUNITS
C      CHARACTER*20      TEMP(1)
C
      IXX= IX
      IYY= IY
      WCS(1)= 0
      WCS(2)= 1
      WCS(3)= 0
      IFPI= FPI
      WRITE(TEMP(1)(1:18),'(''1 INCH='',I6,''' FEET''')'') IFPI
      CALL VUINCH(VUNITS)
      CALL PLTEXT(IXX,IYY,3,0,TEMP(1),18,WCS)
      IYY= IYY+ (VUNITS*.33)
      IY1= IYY+ (VUNITS*.1)
      IY2= IYY- (VUNITS*.1)
      IX1= IXX+ VUNITS
      CALL MOVETO(IXX,IYY)
      CALL LINETO(IX1,IYY,WCS)
      CALL MOVETO(IXX,IY1)
      CALL LINETO(IXX,IY2,WCS)
      CALL MOVETO(IX1,IY1)

```

```

CALL LINETO(IX1,IY2,WCS)
IXX= IXX+ (VUNITS*.5)
IYY= IYY+ (VUNITS*.33)
CALL MOVETO(IXX,IYY)
IYY= IYY+ (VUNITS*1.5)
CALL LINETO(IXX,IYY,WCS)
IXX= IXX+ (VUNITS*.1)
IYY= IYY- (VUNITS*.2)
CALL LINETO(IXX,IYY,WCS)
CALL PLTEXT(IXX,IYY,5,0,' NORTH',6,WCS)

```

C

```

RETURN
END

```

SUBROUTINE FRAME(TITLE,VLX,VLX,VLX,VUX,VUY,COLOR,STYLE)

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

ARGUMENT	TYPE	DESCRIPTION
TITLE(1)	CHARACTER*40	MAP TITLE
VLX	INTEGER*4	VIRTUAL X LOWER LEFT COORDINATE
VLX	INTEGER*4	" Y " " "
VUX	INTEGER*4	" X UPPER RIGHT "
VUY	INTEGER*4	" Y " " "
COLOR	INTEGER*4	FRAME LINE COLOR
STYLE	"	" " STYLE

```

INTEGER VLX,VLX,VLX,VUX,VUY,COLOR,STYLE
CHARACTER*40 TITLE(1)

```

PROGRAM VARIABLES

```

INTEGER WCS(3), IX,IY,SIZE

```

LOAD LINE COLOR AND STYLE INTO WCS ARRAY

```

WCS(1)= 0
WCS(2)= COLOR
WCS(3)= STYLE

```

DRAW FRAME AS DEFINED BY VIRTUAL COORDINATE LIMITS

```

CALL MOVETO(VLX,VLX)
CALL LINETO(VLX,VUY,WCS)
CALL LINETO(VUX,VUY,WCS)
CALL LINETO(VUX,VLX,WCS)
CALL LINETO(VLX,VLX,WCS)

```

```

CALL CHRSTZ(6,SIZE)
IX= VLX+(5*SIZE)
IY= VUY+(.5*SIZE)
CALL PLTEXT(IX,IY,6,0,TITLE(1),40,WCS)

```

```

RETURN
END
SUBROUTINE CORPLT(FPI,TCP,STAMPS)

```

```

CHARACTER*15 STAMPS(2)

```

```

REAL      FPI, TCP(5), TEMP(5), VUNITS

CALL PLINIT(0, 0, 2300, 1700)

CALL VUINCH(VUNITS)

C  CONVERT THE CORRIDOR TO INCHES
CORWID =TCP(4)
CORDIR =TCP(5)
DO 5 I=1, 3, 1
    TEMP(I)=TCP(I)
    IF (TEMP(I) .GT. 0.) THEN
        TEMP(I)=TEMP(I)/FPI
        TEMP(I)=TEMP(I)*VUNITS
    ENDIF
5  CONTINUE
TEMP(4)= (1800./FPI) * VUNITS

C  DRAW THE CORRIDOR
CALL DRWCOR(1100, 875, TEMP, 4, CORWID, CORDIR)

C  DRAW THE LEGEND
IX= 1900
IY= 1950
CALL LEGEND(IX, IY, TCP, STAMPS)

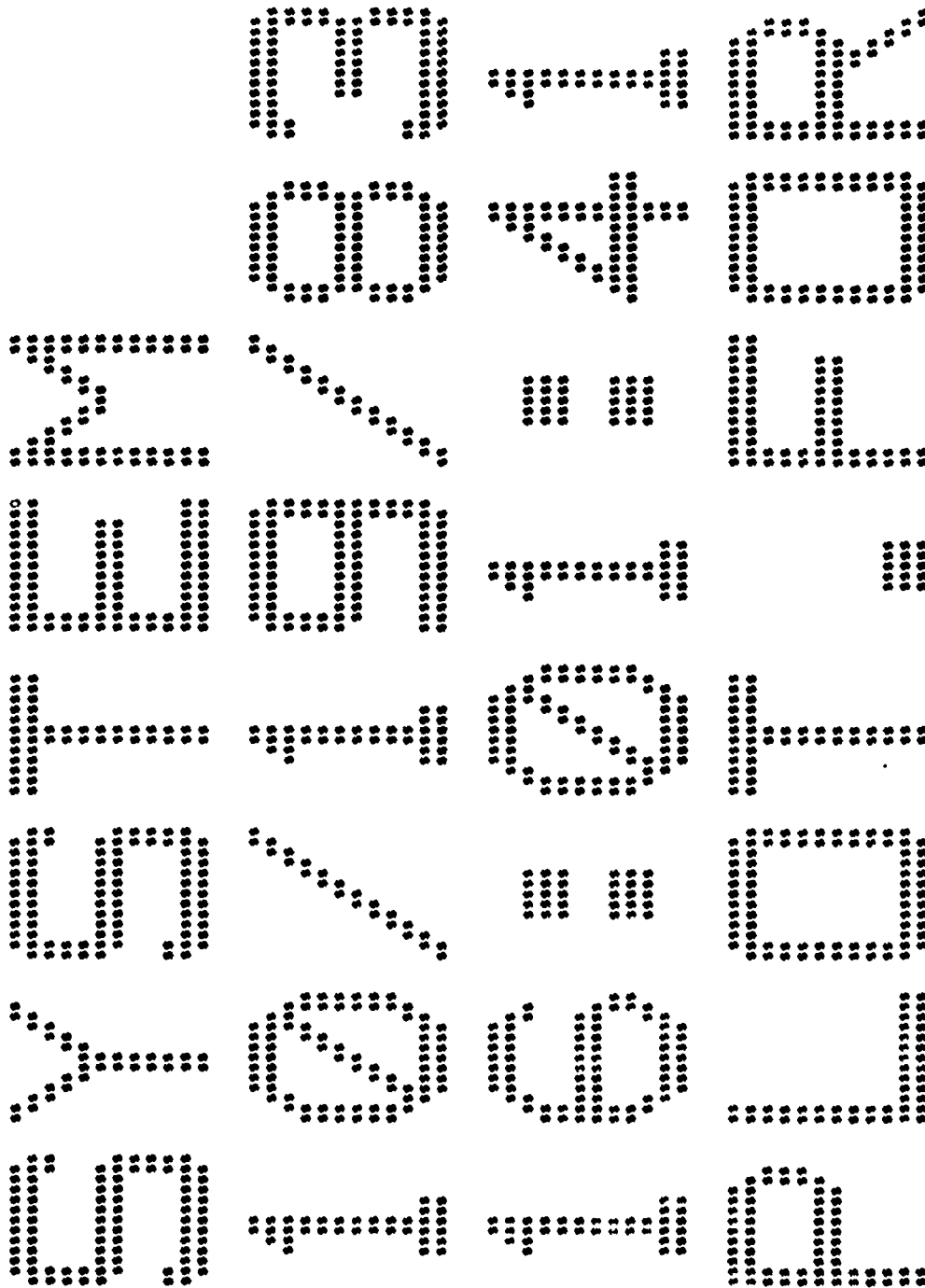
C  DRAW THE SCALE
IX= 2150
IY= 0
CALL SCLPLT(IX, IY, FPI)

C
C  FINISH THE PLOTTING
CALL PLDONE

RETURN
END

```

User - system Date - 10/19/83 Time - 16:01:41 Filename - plot.for



```

C      SUBROUTINE PLINIT(VLX,VLY,VUX,VUY)
C
C      PLOTTER INITIALIZATION SUBROUTINE
C
C      ARGUMENTS      TYPE      DESCRIPTION
C      VLX      INTEGER*4      VIRTUAL LOWER LEFT X COORDINATE
C      VLY      INTEGER*4      " " " Y "
C      VUX      INTEGER*4      " UPPER RIGHT X "
C      VUY      INTEGER*4      " " " Y "
C
C      INTEGER*4      VLX, VLY, VUX, VUY
C
C
C      COMMON/PLTVAR/ CHAN, PSTAT(10), SCALEX, SCALEY
C      COMMON/PLTLOC/ VXY
C
C      INTEGER*4      CHAN, PSTAT,VXY(4)
C      REAL*4      SCALEX, SCALEY
C
C      COMMON BLOCK VARIABLES
C
C      VARIABLE      TYPE      DESCRIPTION
C      CHAN      INTEGER*4      CHANNEL ASSIGNED TO PLOTTER BY CROMIX
C      PSTAT(10)      INTEGER*4      STATUS INFORMATION FOR PLOTTER
C      SCALEX      REAL*4      X SCALE : (PLOT UNITS/VIRTUAL UNITS)
C      "
C      VXY(4)      INTEGER*4      VXY(1)= VLX
C      "      "      VXY(2)= VLY
C      "      "      VXY(3)= VUX
C      "      "      VXY(4)= VUY
C
C      PROGRAM VARIABLES
C
C      INTEGER      IWX,IWY
C      REAL*4      DX, DY, ASPECT, VSPECT
C
C      LOAD /PLTLOC/ COMMON BLOCK
C
C      VXY(1)= VLX
C      VXY(2)= VLY
C      VXY(3)= VUX
C      VXY(4)= VUY
C
C      CHECK Y/X ASPECT RATIO OF VIRTUAL LIMITS. IF THE RATIO IS GREATER
C      THAN THE ASPECT RATIO OF THE PLOTTER (11/17)), THE SCALE FACTORS MUST
C      BE DEPENDENT ON THE Y DIMENSION OF THE PLOT
C
C      DX= FLOAT(VUX-VLX)
C      DY= FLOAT(VUY-VLY)
C      VSPECT= DY/DX
C      ASPECT= 17.0/30.0
C
C      IF( VSPECT .GT. ASPECT ) THEN
C          SCALEY= 1700.0/DY
C          SCALEX= SCALEY
C      ELSE
C          SCALEX= 3000.0/DX

```

```

        SCALEY= SCALEX
    ENDIF

C
C  OPEN A CHANNEL TO THE HOUSTON PLOTTER WITH A CALL TO THE ASSEMBLY LEVEL
C  SUBROUTINE OPNDEV
C
C      CALL OPNDEV('/DEV/PLOT',CHAN)
C
C  HOME THE PLOTTER, SET ABSOLUTE MODE, SET .005-INCH INCREMENTS
C
C      CALL OUTDEV(' ;:HA EC5',8,CHAN)
C
C  SET THE ORIGIN 1 INCH IN AND UP FROM THE HOME POSITION
C
C      CALL OUTDEV('U 100,100 ;:O',13,CHAN)
C
C      RETURN
C      END

        SUBROUTINE CHRSTZ(HT,SIZE)
C
C  CHRSTZ RETURNS THE SIZE OF PLOTTED CHARACTERS IN VIRTUAL UNITS GIVEN
C  THEN HEIGHT IN PLOTTER UNITS
C
C      ARGUMENT          TYPE          DESCRIPTION
C      HT                INTEGER*4      HEIGHT OF CHARACTERS IN PLOTTER UNITS
C      SIZE              "              SIZE OF CHARACTERS IN VIRTUAL UNITS
C
C      INTEGER HT,SIZE
C
C
C      COMMON/PLTVAR/ CHAN, PSTAT(10), SCALEX, SCALEY
C
C      INTEGER*4        CHAN, PSTAT
C      REAL*4           SCALEX, SCALEY
C
C  COMMON BLOCK VARIABLES
C
C      VARIABLE          TYPE          DESCRIPTION
C      CHAN              INTEGER*4      CHANNEL ASSIGNED TO PLOTTER BY CROMIX
C      PSTAT(10)         INTEGER*4      STATUS INFORMATION FOR PLOTTER
C      SCALEX            REAL*4         X SCALE : (PLOT UNITS/VIRTUAL UNITS)
C      "
C
C  PROGRAM VARIABLES
C
C      REAL      HEIGHT, PUNITS
C
C
C  SINCE THE HOUSTON IS SET TO .005 INCHES IN PLINIT, THE CHARACTER SIZE
C  IN INCHES IS: SIZE(IN)= .035+(HT*.035)
C
C      HEIGHT= .035+ (HT*.035)
C      PUNITS= HEIGHT/.005
C      SIZE= PUNITS/SCALEY
C

```

RETURN
END

SUBROUTINE PLREAD (STRING, CHARS)

```

C
C PLREAD READS OUTPUT FROM THE PLOTTER
C
C ARGUMENT      TYPE      DESCRIPTION
C STRING        CHARACTER  UP TO 80 BYTES RECEIVED FROM THE PLOTTER
C CHARS         INTEGER*2  NUMBER OF BYTES SENT BY THE PLOTTER
C
C CHARACTER*(*)  STRING
C INTEGER*4      CHARS
C
C COMMON BLOCK /PLTVAR/ MUST HAVE BEEN INITIALIZED BY A CALL TO SUBROUTINE
C PLINIT
C
C COMMON/PLTVAR/ CHAN, PSTAT(10), SCALEX, SCALEY
C
C INTEGER*4      CHAN, PSTAT
C REAL*4         SCALEX, SCALEY
C
C CHARACTER*99 STR, DUM
C
C STR=' '
C OPEN(8, FILE=' /DEV/PLOT')
C CALL OPNDEV(' /DEV/PLOT', ICHAN)
C WRITE(0, '( " PRESS RETURN TO BEGIN DIGITIZING" )')
C READ(0, '(A99)') STR
C
C CALL OUTDEV(' ;EL ED', 7, CHAN)
C ICNT= 16
C CALL INDEV(STR, ICNT, ICHAN)
C WRITE(8, '( " ;EL ED" )')
C READ(8, '(A99)') STR
C WRITE(0, '(I3/A99)') ICNT, STR
C
C WRITE(0, '( " PRESS RETURN TO BEGIN DIGITIZING" )')
C READ(0, '(A99)') STR
C
C CALL OUTDEV(' ;EL ED', 7, CHAN)
C WRITE(8, '( " ;EL ED" )')
C READ(8, '(A99)') STR
C ICNT= 16
C CALL INDEV(STR, ICNT, ICHAN)
C WRITE(0, '(I3/A99)') ICNT, STR
C
C CLOSE(8)
C RETURN
C END

```

SUBROUTINE DELAY

```

DO 10 I=1,30000
  X=X*1.1+X*(-1.1)
10 CONTINUE
  RETURN
  END

```

SUBROUTINE PLDONE

```

C
C PLDONE RESETS THE PLOTTER AND CLOSES THE PLOT CHANNEL
C
C
C COMMON BLOCK /PLTVAR/ MUST HAVE BEEN INITIALIZED BY A CALL TO SUBROUTINE
C PLINIT
C
  COMMON/PLTVAR/ CHAN, PSTAT(10), SCALEX, SCALEY
C
  INTEGER*4      CHAN, PSTAT
  REAL*4         SCALEX, SCALEY
C
C PROGRAM VARIABLES
C
  CHARACTER*5     COMMND(1)
C
C ASSEMBLE THE COMMAND TO RESET THE PLOTTER
C
  WRITE(COMMND(1)(1:5),100)
100 FORMAT( ' Z ' )
  CALL OUTDEV(COMMND(1),5,CHAN)
C
C CLOSE THE CHANNEL
C
  CALL CLSDEV(CHAN)
C
  RETURN
  END

```

SUBROUTINE LINETO(VX,VY,LATT)

```

C
C LINETO WILL DRAW A LINE TO THE COORDINATES VX,VY WITH LINE ATTRIBUTES
C AS SPECIFIED IN ARRAY LATT
C
C ARGUMENT      TYPE      DESCRIPTION
C   VX          INTEGER*4  VIRTUAL X COORDINATE OF DRAW DESTINATION
C   VY          INTEGER*4  "      Y      "      "      "
C   LATT(3)     INTEGER*4  LATT(1): LINE WEIGHT (NOT IMPLEMENTED)
C                        LATT(2): LINE COLOR
C                        LATT(3): LINE STYLE
C
C   INTEGER*4      VX, VY, LATT(3)
C
C COMMON BLOCK /PLTVAR/ MUST HAVE BEEN INITIALIZED BY A CALL TO SUBROUTINE
C PLINIT
C
  COMMON/PLTVAR/ CHAN, PSTAT(10), SCALEX, SCALEY

```

```

      INTEGER*4      CHAN, PSTAT
      REAL*4         SCALEX, SCALEY

C
      COMMON/PLTLOC/ VXY

C
      INTEGER*4      VXY(4)

C
C   COMMON BLOCK VARIABLES
C
C      VARIABLE      TYPE      DESCRIPTION
C      VXY(4)        INTEGER*4  VXY(1)= VLX
C                               VXY(2)= VLY
C                               VXY(3)= VUX
C                               VXY(4)= VUY
C
C   PROGRAM VARIABLES
C
C      CHARACTER*22   COMMND(1)
C      INTEGER*4      COLOR, STYLE, PX, PY
C      REAL*4         RPX, RPY

C
C      COLOR= LATT(2)
C      STYLE= LATT(3)
C   ADJUST ORIGIN AND
C   MULTIPLY VX, VY BY SCALE FACTORS TO OBTAIN PLOTTER UNITS
C
C      RPX= FLOAT(VX-VXY(1)) * SCALEX
C      RPY= FLOAT(VY-VXY(2)) * SCALEY
C      PX=  INT(RPX)
C      PY=  INT(RPY)

C
C   ASSEMBLE COMMAND FOR A MOVE WITH THE PEN DOWN
C
C      WRITE(COMMND(1)(1:22),100) COLOR, STYLE, PX, PY
100  FORMAT(' P',I1,' L',I1,' D ',I5,',',I5,' ')
C
C      CALL OUTDEV(COMMND(1),22,CHAN)

C
C      RETURN
C      END

C
C      SUBROUTINE MOVETO(VX,VY)
C
C   MOVETO WILL MOVE THE PEN TO THE COORDINATES VX,VY
C
C      ARGUMENT      TYPE      DESCRIPTION
C      VX            INTEGER*4  VIRTUAL X CORDINATE OF MOVE DESTINATION
C      VY            INTEGER*4  "      Y      "      "      "
C
C      INTEGER*4      VX, VY

C
C   COMMON BLOCK /PLTVAR/ MUST HAVE BEEN INITIALIZED BY A CALL TO SUBROUTINE
C   PLINIT
C
C      COMMON/PLTVAR/ CHAN, PSTAT(10), SCALEX, SCALEY

```

```

      INTEGER*4      CHAN, PSTAT
      REAL*4         SCALEX, SCALEY

C
      COMMON/PLTLOC/ VXY

C
      INTEGER*4      VXY(4)

C
      COMMON BLOCK VARIABLES

C
      VARIABLE      TYPE      DESCRIPTION
      VXY(4)        INTEGER*4  VXY(1)= VLX
C                                     VXY(2)= VLY
C                                     VXY(3)= VUX
C                                     VXY(4)= VUY
C
      PROGRAM VARIABLES

C
      CHARACTER*16   COMMND(1)
      INTEGER*4      PX, PY
      REAL*4         RPX, RPX

C
      ADJUST ORIGIN AND
      MULTIPLY VX, VY BY SCALE FACTORS TO OBTAIN PLOTTER UNITS

C
      RPX= FLOAT(VX-VXY(1)) * SCALEX
      RPY= FLOAT(VY-VXY(2)) * SCALEY
      PX=  INT(RPX)
      PY=  INT(RPY)

C
      ASSEMBLE THE COMMAND TO MOVE WITH THE PEN UP

C
      WRITE(COMMND(1)(1:16),100) PX, PY
100  FORMAT(' U ',I5,',',I5,',')
      CALL OUTDEV(COMMND(1),16,CHAN)

C
      RETURN
      END

      SUBROUTINE MARKER(MID,VX,VY,LATT,SCALE,ANGLE)

C
      MARKER WILL DRAW A MARKER AT COORDINATES VX, VY

C
      ARGUMENT      TYPE      DESCRIPTION
      MID           INTEGER*4  MARKER TYPE (SEE HOUSTON PLOTTER DOC)
      VX            INTEGER*4  VIRTUAL X CORDINATE OF MOVE DESTINATION
      VY            INTEGER*4  " " " " " " "
      LATT(3)       INTEGER*4  LATT(1): LINE WEIGHT (NOT IMPLEMENTED)
C                                     LATT(2): LINE COLOR
C                                     LATT(3): LINE STYLE
      SCALE         REAL*4     HEIGHT SPECIFIER
      ANGLE         INTEGER*4  MARKER ORIENTATION (NOT IMPLEMENTED)

C
      INTEGER*4      MID, VX, VY, LATT(3), ANGLE
      REAL*4         SCALE

C
      COMMON BLOCK /PLTVAR/ MUST HAVE BEEN INITIALIZED BY A CALL TO SUBROUTINE
      PLINIT

```

```

COMMON/PLTVAR/ CHAN, PSTAT(10), SCALEX, SCALEY
C
  INTEGER*4      CHAN, PSTAT
  REAL*4         SCALEX, SCALEY

```

```

C
C
C PROGRAM VARIABLES
C

```

```

  CHARACTER*12    COMMND(1)
  INTEGER*4       COLOR, STYLE, HT

```

```

  COLOR= LATT(2)
  STYLE= LATT(3)

```

```

C
C SET MARKER HEIGHT
C

```

```

  HT= INT(SCALE)
  IF( HT .LE. 0 ) HT=1
  IF( HT .GT. 5 ) HT=5

```

```

C
C MOVE TO VX, VY
C

```

```

  CALL MOVETO(VX,VY)

```

```

C
C ASSEMBLE THE COMMAND TO DRAW THE MARKER
C

```

```

  WRITE(COMMND(1)(1:12),100) COLOR, STYLE, HT, MID
100  FORMAT(' P',I1,' L',I1,' M',I1,I1,I1,I1)
  CALL OUTDEV(COMMND(1),12,CHAN)

```

```

C
  RETURN
  END

```

```

SUBROUTINE PLTEXT(VX,VY,HT,ANG,STRING,NCHAR,LATT)

```

```

C
C PLTEXT DRAWS TEXT PASSED IN 'STRING', BEGINNING AT VX,VY AND AT ANGLE ANG
C

```

ARGUMENT	TYPE	DESCRIPTION
VX	INTEGER*4	VIRTUAL X CORDINATE OF DRAW DESTINATION
VY	INTEGER*4	" Y " " " "
HT	INTEGER*4	HEIGHT OF TEXT IN VIRTUAL UNITS
ANG	INTEGER*4	ANGLE OF TEXT (0-360)
STRING	CHARACTER*(*)	ARRAY CONTAINING TEXT
NCHAR	INTEGER*4	NUMBER OF CHARACTERS TO DRAW
LATT(3)	INTEGER*4	LATT(1): LINE WEIGHT (NOT IMPLEMENTED)
		LATT(2): LINE COLOR
		LATT(3): LINE STYLE

```

  INTEGER*4      VX, VY, HT, ANG, NCHAR, LATT(3)
  CHARACTER*(*)  STRING

```

```

C
C COMMON BLOCK /PLTVAR/ MUST HAVE BEEN INITIALIZED BY A CALL TO SUBROUTINE
C PLINIT
C

```

```

COMMON/PLTVAR/ CHAN, PSTAT(10), SCALEX, SCALEY

```

```

      INTEGER*4      CHAN, PSTAT
      REAL*4        SCALEX, SCALEY

C
C
C PROGRAM VARIABLES
C
      INTEGER*4      PH, SX, SY, COLOR, STYLE
      REAL*4        RADS
      CHARACTER*30   COMMND(1)

C
C
      COLOR= LATT(2)
      STYLE= LATT(3)
      RADS= FLOAT(ANG)/57.35
      SX= NINT( 1000.*COS(RADS) )
      SY= NINT( 1000.*SIN(RADS) )
      PH= HT
      CALL MOVETO(VX,VY)
      WRITE(COMMND(1)(1:30),100) COLOR,STYLE,PH,SX,SY
100  FORMAT(' P',I1,' L',I1,' S(S',I5,' X',I5,' Y',I5,' )' )
      ICNT= 30
      CALL OUTDEV(COMMND(1),ICNT,CHAN)
      CALL OUTDEV(STRING,NCHAR,CHAN)
      ICNT= 2
      CALL OUTDEV('_ ',ICNT,CHAN)

C
      RETURN
      END

      SUBROUTINE CIRCLE(VX,VY,VR,LATT)
C
C CIRCLE DRAWS A CIRCLE AT VX, VY WITH RADIUS VR
C
C ARGUMENT      TYPE      DESCRIPTION
C      VX      INTEGER*4    VIRTUAL X CORDINATE OF DRAW DESTINATION
C      VY      INTEGER*4    "      Y      "      "      "
C      VR      INTEGER*3    VIRTUAL RADIUS OF CIRCLE
C      LATT(3)  INTEGER*4    LATT(1): LINE WEIGHT (NOT IMPLEMENTED)
C                               LATT(2): LINE COLOR
C                               LATT(3): LINE STYLE
C
C      INTEGER*4      VX, VY, VR, LATT(3)

C
C COMMON BLOCK /PLTVAR/ MUST HAVE BEEN INITIALIZED BY A CALL TO SUBROUTINE
C PLINIT
C
      COMMON/PLTVAR/ CHAN, PSTAT(10), SCALEX, SCALEY

C
      INTEGER*4      CHAN, PSTAT
      REAL*4        SCALEX, SCALEY

C
C PROGRAM VARIABLES
C
      CHARACTER*27   COMMND(1)
      INTEGER*4      PX, PY, PR, COLOR, STYLE

```



```

$BIGCODE
$SEGMENT XPRDBSEG
      SUBROUTINE PROCD(ITR, PUNIT1, PUNIT2, PFILE1, PHFILE, SMUNIT, SMFILE)

C*****
C  THIS SUBROUTINE MANAGES THE PROCEDURE DATA BASE
C  VARIABLES PASSED:
C
C      ITR      -  INTERACTIVE TERMINAL READ UNIT
C      PUNIT1   -  UNIT # TO OPEN THE PROCEDURE DIRECTORY FILE ON
C      PUNIT2   -  UNIT # TO OPEN THE PROCEDURE FILE ON
C      PFILE1   -  DIRECTORY FILE NAME
C      PHFILE   -  PROCEDURE HELP FILE NAME
C      SMUNIT   -  UNIT # TO OPEN MENU FILE (SMFILE) ON
C      SMFILE   -  MENU FILE NAME
C*****

      CHARACTER*1  CMD(1)
      CHARACTER*7  PFILE1, PHFILE, SMFILE

      INTEGER      ITR, PUNIT1, PUNIT2, SMUNIT, RC(2, 3)

C  DISPLAY THE MAIN MENU AND INPUT USER SELECTION
1  CALL MENUSV(SMFILE, 140, RC, 2, SMUNIT)
2  CMD(1)=' '
   CALL MENURD(RC, 2, 1, 1, CMD, ITR)

C  CHECK FOR A VALID INPUT
   IF (INDEX('1234X', CMD(1)) .EQ. 0) THEN
       CALL MESS(11, RC(2, 1), RC(2, 2), RC(2, 3), 7)
       GOTO 2
   ENDIF

C  USER SELECTED TO VIEW THE HELP FILES
   IF (CMD(1) .EQ. '1') CALL PHELP(ITR, PUNIT1, PHFILE, SMUNIT, SMFILE)

C  USER SELECTED TO DELETE A FILE FROM THE DATA BASE
   IF (CMD(1) .EQ. '2') CALL PDEL(ITR, PUNIT1, PUNIT2, PFILE1, SMUNIT,
*                               SMFILE)

C  USER SELECTED TO ADD TO THE DATA BASE
   IF (CMD(1) .EQ. '3') CALL PADD(ITR, PUNIT1, PUNIT2, PFILE1, SMUNIT,
*                               SMFILE)

C  USER SELECTED TO SEARCH THE DATABASE
   IF (CMD(1) .EQ. '4') CALL PSEAR(ITR, PUNIT1, PUNIT2, PFILE1, SMUNIT,
*                               SMFILE)

C  USER SELECTED TO RETURN
   IF (CMD(1) .EQ. 'X') RETURN

   GOTO 1
END
      SUBROUTINE PHELP(ITR, PUNIT, PHFILE, SMUNIT, SMFILE)
C*****
C  THIS SUBROUTINE DISPLAYS THE HELP FILE FOR THE PROCEDURE DATA BASE
C  VARIABLES PASSED:
C
C      ITR      -  INTERACTIVE TERMINAL READ UNIT
C      PUNIT    -  UNIT # TO OPEN THE HELP FILE (PHFILE) ON

```

```

C      PHFILE - PROCEDURE HELP FILE NAME
C      SMUNIT - UNIT # TO OPEN MENU FILE (SMFILE) ON
C      SMFILE - MENU FILE NAME
C*****
      CHARACTER*80 TLINE,LINE
      CHARACTER*7  PHFILE,SMFILE
      CHARACTER*1  CMD(1)

      INTEGER      ITR,PUNIT,SMUNIT,RC(3,3)

      LOGICAL      FLAG

C      DISPLAY THE MAIN MENU AND INPUT THE USER SELECTION
1      CALL MENUSV(SMFILE,145,RC,3,SMUNIT)
2      CMD(1)=' '
      CALL MENURD(RC,3,1,1,CMD,ITR)

C      CHECK FOR A VALID INPUT
      IF (INDEX('123X',CMD(1)) .EQ. 0) THEN
          CALL MESS(11,RC(2,1),RC(2,2),RC(2,3),7)
          GOTO 2
      ENDIF

C      USER SELECTED TO RETURN
      IF (CMD(1) .EQ. 'X') RETURN

C      CHECK TO SEE THAT THE HELP FILE EXISTS, IF NOT DISPLAY ERROR MESSAGE
      INQUIRE (FILE=PHFILE,EXIST=FLAG)
      IF (.NOT. FLAG) THEN
          CALL MESS(15,RC(2,1),RC(2,2),RC(2,3),7)
          GOTO 2
      ENDIF

C      CLEAR THE SCREEN AND OPEN THE HELP FILE
      CALL CLEAR(7,0)
      OPEN (PUNIT,FILE=PHFILE,STATUS='OLD')

C      SET UP THE SEARCH KEY FOR THE FILE TO BE DISPLAYED.

      TLINE(1:80)='*X*'
      TLINE(2:2)=CMD(1)

C      READ IN LINE FROM THE HELP FILE
5      READ(PUNIT,'(A80)') LINE

C      LOOK FOR THE KEY
      IF (TLINE(1:3) .EQ. LINE(1:3)) THEN

C          DETERMINE HOW MANY LINE ARE TO BE DISPLAYED
          READ(LINE(4:5),'(I2)') IG
          J=0

C          START TO DISPLAY THE LINES ON THE SCREEN
          DO 20 I=1,IG,1
              READ(PUNIT,'(A80)') LINE
              J=J+1
              CALL MENUDR(LINE,J,1,2,0,1,1)

C          CAN ONLY DISPLAY 22 LINES ON THE SCREEN AT ONCE
          IF (J .EQ. 22) THEN

```

```

C          END-OF-FILE REACHED DISPLAY MESSAGE
          IF (I .EQ. IG) THEN
            CALL MESS(19, RC(3, 1), RC(3, 2), RC(3, 3), 7)
            READ(ITR, '(A1)') CMD(1)
            CLOSE (PUNIT)
            GOTO 1
          ELSE

C          MORE TO BE DISPLAY (PRESS RETURN TO CONTINUE)
            CALL MESS(16, RC(3, 1), RC(3, 2), RC(3, 3), 7)
            READ(ITR, '(A1)') CMD(1)
            CALL CLEAR(7, 0)
            J=0
          ENDIF
          ENDIF

20        CONTINUE

C          END-OF-FILE REACHED
            CALL MESS(19, RC(3, 1), RC(3, 2), RC(3, 3), 7)
            READ(ITR, '(A1)') CMD(1)
            CLOSE (PUNIT)
            GOTO 1
          ENDIF

```

```

GOTO 5
END
SUBROUTINE PADD(ITR, PUNIT1, PUNIT2, PFILE1, SMUNIT, SMFILE)

```

```

C*****
C  THIS SUBROUTINE ADDS ENTRIES TO THE PROCEDURE DATA BASE.  THERE IS A FILE
C  THAT CONTAINS A LIST OF ALL THE FILES ON THE SYSTEM AND NAME OF THE
C  PROCEDURES ASSOCIATED WITH THOSE FILES
C    VARIABLES PASSED:
C
C    ITR      -  INTERACTIVE TERMINAL READ UNIT
C    PUNIT1   -  UNIT # TO OPEN PFILE1 ON
C    PUNIT2   -  UNIT # TO OPEN THE PROCEDURE FILE ON
C    PFILE1   -  DIRECTORY FILE NAME
C    SMUNIT   -  UNIT # TO OPEN MENU FILE (SMFILE) ON
C    SMFILE   -  MENU FILE NAME
C*****

```

```

CHARACTER*80  LINE
CHARACTER*40  PKEY(1), CIN, TEXT(2)
CHARACTER*7   PFILE1, EFILE, SMFILE, PFILE2
CHARACTER*1   CMD(1)

INTEGER       ITR, PUNIT1, PUNIT2, SMUNIT, RC(5, 3), ST(3)

LOGICAL       FLAG1, FLAG2

DATA ST/0, 0, 0/

ITT=-1
PKEY(1)(1:40)=' '
TEXT(1)(1:40)='ENTER "ABORT/" TO ABORT THE FILE'
TEXT(2)(1:40)='ENTER "END/" TO SAVE THE FILE'
FLAG2=.FALSE.

```

```

C   DISPLAY THE MAIN MENU
100 CALL MENUSV(SMFILE,146,RC,5,SMUNIT)

    IF (FLAG2) THEN
        ST(3)=0
        IF (PKEY(1) .NE. ' ')
            *   CALL MENUWR(RC,5,1,1,PKEY,0,1,ST)
                GOTO 15
            ENDIF

C   INPUT THE PROCEDURE FILE NAME
5   CALL MENURD(RC,5,1,1,PKEY,ITR)
    FLAG2 =.FALSE.

C   SCANNING THE COMMAND INPUT LINE
15  CMD(1)=' '
    CALL MENURD(RC,5,2,2,CMD,ITT)

C   USER SELECTED TO GO BACK TO THE PROCEDURE INPUT LINE
    IF (CMD(1) .EQ. ' ') THEN
        ST(3)=1
        CALL MENUWR(RC,5,2,2,CMD,0,1,ST)
        CALL MESS(4,RC(5,1),RC(5,2),RC(5,3),1)
        GOTO 5
    ENDIF

C   USER SELECTED TO RETURN
    IF (CMD(1) .EQ. 'X') RETURN

C   INVALID INPUT
    IF (CMD(1) .NE. 'C') GOTO 15

C   A BLANK PROCEDURE NAME IS NOT ALLOWED
    IF (PKEY(1) .EQ. ' ') THEN
        CALL MESS(20,RC(5,1),RC(5,2),RC(5,3),7)
        GOTO 15
    ENDIF

C   CHECK IF THE DIRECTORY FILE EXISTS
    INQUIRE (FILE=PFILE1,EXIST=FLAG1)

C   CREATE THE DIRECTORY FILE
    IF (.NOT. FLAG1) THEN
        OPEN (PUNIT1,FILE=PFILE1,STATUS='NEW',FORM='UNFORMATTED',
            *   ACCESS='DIRECT',RECL=48)
                NREC=1

C   WRITE THE HEADER RECORD
                WRITE(PUNIT1,REC=1) NREC
                WRITE(PUNIT1,REC=2) NREC
                CLOSE (PUNIT1)
            ENDIF

C   CHECK TO SEE IF THE PROCEDURE IS ALREADY IN THE DATA BASE
    OPEN (PUNIT1,FILE=PFILE1,STATUS='OLD',ACCESS='DIRECT',
        *   FORM='UNFORMATTED',RECL=48)

C   FIND OUT HOW MANY FILES ARE ALREADY IN THE DATA BASE
    READ(PUNIT1,REC=1) NREC

```

```

C      CHECK TO SEE IF THE PROCEDURE ALREADY EXISTS
      DO 20 I=2,NREC,1
        READ(PUNIT1,REC=I) CIN

C      PROCEDURE FOUND
        IF (CIN .EQ. PKEY(1)) THEN
          CALL MESS(2,RC(5,1),RC(5,2),RC(5,3),7)
          CLOSE (PUNIT1)
          GOTO 15
        ENDIF

20     CONTINUE

C      FIND THE PROC FILE THAT IS TO BE USED.  THE FORM WILL BE PROCXX, WHERE
C      XX =0,1,2,...,99
      DO 25 I=0,99,1
        PFILE2=EFILE(I)
        PFILE2(1:4)='PROC'
        INQUIRE (FILE=PFILE2,EXIST=FLAG1)
        IF (.NOT. FLAG1) GOTO 30

25     CONTINUE

C      TO MANY FILES EXIST
      CALL MESS(21,RC(5,1),RC(5,2),RC(5,3),7)
      GOTO 15

C      DISPLAY THE DIRECTION ABOUT END AND ABORT
30     ST(3)=1
      CALL MENUWR(RC,5,3,4,TEXT,0,1,ST)

C      INPUT THE USERS OPTION
      CMD(1)=' '
      CALL MENURD(RC,5,2,2,CMD,ITT)

C      USER SELECTED RETURN
      IF (CMD(1) .EQ. 'X') THEN
        CLOSE (PUNIT1)
        RETURN
      ENDIF

C      OPEN THE PROCEDURE FILE, THE USER IS GOING TO CREATE
      OPEN (PUNIT2,FILE=PFILE2,STATUS='NEW')

C      CLEAR THE SCCREEN AND TURN THE CURSOR ON
      CALL CLEAR(7,0)
      CALL ONOFF(1)

C      READ THE USERS LINE OF TEXT
35     READ(ITR,'(A80)') LINE

C      USER DECIDED NOT TO SAVE THE PROCEDURE FILE
      IF (LINE .EQ. 'ABORT//') THEN
        CLOSE (PUNIT1)
        CLOSE (PUNIT2,STATUS='DELETE')
        FLAG2=.TRUE.
        CALL ONOFF(0)
        GOTO 100
      ENDIF

C      USER WANTS TO SAVE THE PROCEDURE FILE
      IF (LINE .EQ. 'END//') THEN

```

```

CLOSE (PUNIT2)
NREC=NREC+1
WRITE(PUNIT1,REC=NREC) PKEY,PFILE2
III=NREC+1
WRITE(PUNIT1,REC=III) PKEY,PFILE2
WRITE(PUNIT1,REC=1) NREC
CLOSE (PUNIT1)
CALL ONOFF(0)
PKEY(1)(1:40)=' '
FLAG2=.TRUE.
GOTO 100
ENDIF

```

```

C   WRITE THE LINE TO THE PROCEDURE FILE PROCXX
WRITE(PUNIT2,'(A80)') LINE
GOTO 35

```

```

END
SUBROUTINE PSEAR(ITR,PUNIT1,PUNIT2,PFILE1,SMUNIT,SMFILE)

```

```

C*****
C   THIS SUBROUTINE SEARCHES THE PROCEDURE DATA BASE AND WILL DISPLAY THE FILE
C   THAT THE USER SELECTS
C   VARIABLES PASSED:
C
C   ITR      -   INTERACTIVE TERMINAL READ UNIT
C   PUNIT1   -   UNIT # TO OPEN PFILE1 ON
C   PUNIT2   -   UNIT # TO OPEN PROCEDURE FILE ON
C   PFILE1   -   PROCEDURE DIRECTORY FILE
C   SMUNIT   -   UNIT # TO OPEN MENU FILE (SMFILE) ON
C   SMFILE   -   MENU FILE NAME
C*****

```

```

CHARACTER*70 OUT(19)
CHARACTER*40 PKEY
CHARACTER*7  PFILE1,PFILE2,EFILE,SMFILE
CHARACTER*2  CMD(1),TEMP

```

```

LOGICAL      FLAG1,FLAG2,FLAG3

```

```

INTEGER      ITR,JJ,CNT,PUNIT1,PUNIT2,RC(20,3),SMUNIT,NREC,ST(3)

```

```

DATA ST/0,0,0/

```

```

C   DISPLAY THE MAIN MENU
100 CALL MENUSV(SMFILE,147,RC,20,SMUNIT)

C   CHECK TO SEE THAT THE PROCEDURE DATA BASE EXISTS
INQUIRE (FILE=PFILE1,EXIST=FLAG1)

C   PROCEDURE DIRECTORY FILE NOT FOUND
IF (.NOT. FLAG1) THEN
    CALL MESS(18,RC(20,1),RC(20,2),RC(20,3),7)
    CMD(1)=' '
    CALL MENURD(RC,20,1,1,CMD,ITR)
    IF (CMD(1).EQ. 'X ') RETURN
    GOTO 5
ENDIF

```

```

C      OPEN THE PROCEDURE DIRECTORY FILE
      OPEN (PUNIT1, FILE=PFILE1, STATUS='OLD', ACCESS='DIRECT',
*       FORM='UNFORMATTED', RECL=48)

C      FOUND OUT HOW MANY RECORDS ARE IN THE FILE
      READ(PUNIT1, REC=1) NREC
      FLAG1=.TRUE.
      FLAG2=.TRUE.
15     CNT=2

C      PROCESS THE DIRECTORY FILE AND GET THE PROCEDURE NAMES
      DO 20 I=2, NREC, 1
        READ(PUNIT1, REC=I) PKEY, PFILE2
        FLAG2=.FALSE.
        FLAG1=.FALSE.
        OUT(CNT)(1:70)=' '
        WRITE(OUT(CNT)(2:3), '(I2)') I
        OUT(CNT)(6:70)=PKEY

C      CAN ONLY DISPLAY 19 PROCEDURE NAMES ON THE SCREEN AT A TIME
      IF (CNT .EQ. 19) THEN
        CALL MENUWR(RC, 20, 2, 19, OUT, 0, 1, ST)

C      INPUT THE USER SELECTION
45     CMD(1)=' '
        CALL MENURD(RC, 20, 1, 1, CMD, ITR)

C      USER SELECTED TO RETURN
      IF (CMD(1) .EQ. 'X ') THEN
        CLOSE (PUNIT1)
        RETURN
      ENDIF

C      USER SELECTED TO CONTINUE VIEWING THE LIST
      IF (CMD(1) .EQ. 'C ') THEN
        FLAG1=.TRUE.
        CALL MENUSV(SMFILE, 147, RC, 20, SMUNIT)
        CNT=2
        GOTO 20
      ENDIF

C      VALIDATE THE USER'S INPUT
      READ(CMD(1), '(BN, I2)', ERR=45) JJ
      IF ((JJ .LE. 1).OR.(JJ .GT. NREC)) GOTO 45
      READ(PUNIT1, REC=JJ) PKEY, PFILE2
      INQUIRE (FILE=PFILE2, EXIST=FLAG3)
      IF (.NOT. FLAG3) GOTO 45

C      USER SELECTED A VALID PROCEDURE FILE NUMBER SO DISPLAY THE FILE
      CLOSE (PUNIT1)
      CALL PROPRO(ITR, PFILE2, PUNIT1)
      GOTO 100

                                ENDIF

      CNT=CNT+1
20     CONTINUE

      IF ((FLAG1).AND.(.NOT. FLAG2)) GOTO 15
      IF ((FLAG1).AND.(FLAG2))
*     CALL MESS(22, RC(20, 1), RC(20, 2), RC(20, 3), 7)

```

```

READ(PUNIT1,REC=1) PKEY,PFILE2
FLAG2=.FALSE.
FLAG1=.FALSE.
OUT(CNT)(1:70)=' '
WRITE(OUT(CNT)(2:3),'(I2)') I
OUT(CNT)(6:70)=PKEY
IF (ISTART .EQ. -1) ISTART=I
IEND=I

C      CAN DISPLAY ONLY 19 AT A TIME ON THE SCREEN
      IF (CNT .EQ. 19) THEN
          CALL MENUWR(RC,20,2,19,OUT,0,1,ST)

C      INPUT THE USERS SELECTION
45      CMD(1)(1:18)=' '
          CALL MENURD(RC,20,1,1,CMD,ITR)

C      USER SELECTED TO RETURN
          IF (CMD(1) .EQ. 'X') THEN
              CALL PCOMP(PUNIT1,PFILE1)
              CLOSE (PUNIT1)
              RETURN
          ENDIF

C      USER SELECTED TO CONTINUE VIEWING THE LIST
          IF (CMD(1) .EQ. 'C') THEN
              ISTART=-1
              FLAG1=.TRUE.
              CALL MENUSV(SMFILE,148,RC,20,SMUNIT)
              CNT=2
              GOTO 20
          ENDIF

C      DELETE THE FILES FROM THE PROCEDURE DIRECTORY
          INP=CMD(1)
          CALL PROPDE(PUNIT1,PFILE1,INP,ISTART,IEND)
          GOTO 45

                                ENDIF

      CNT=CNT+1
20     CONTINUE

      IF ((FLAG1).AND.(.NOT. FLAG2)) THEN
          CALL PCOMP(PUNIT1,PFILE1)
          GOTO 15
      ENDIF

      IF ((FLAG1).AND.(FLAG2))
      *   CALL MESS(22,RC(20,1),RC(20,2),RC(20,3),7)
          CALL MENUWR(RC,20,2,CNT-1,OUT,0,1,ST)

C      INPUT THE USER SELECTION
35      CMD(1)(1:18)=' '
          CALL MENURD(RC,20,1,1,CMD,ITR)

C      USER SELECTED TO RETURN
          IF (CMD(1) .EQ. 'X') THEN
              CALL PCOMP(PUNIT1,PFILE1)
              CLOSE (PUNIT1)
              RETURN
          ENDIF

```

```

C      NOT 19 PROCEDURE FILES TO DISPLAY
      CALL MENUWR(RC,20,2,CNT-1,OUT,0,1,ST)

C      INPUT THE USERS SELECTION
35     CMD(1)=' '
      CALL MENURD(RC,20,1,1,CMD,ITR)

C      USER SELECTED TO RETURN
      IF (CMD(1) .EQ. 'X ') THEN
                                CLOSE (PUNIT1)
                                RETURN
                                ENDIF

C      USER SELECTED TO CONTINUE VIEWING LIST
      IF (CMD(1) .EQ. 'C ') THEN
                                FLAG1=.TRUE.
                                CALL MENUSV(SMFILE,147,RC,20,SMUNIT)
                                GOTO 15
                                ENDIF

C      CHECK THAT THE USER SELECTED A VALID PROCEDURE FILE
      READ(CMD(1),'(BN,I2)',ERR=35) JJ
      IF ((JJ .LE. 1).OR.(JJ .GT. NREC)) GOTO 35
      READ(PUNIT1,REC=JJ) PKEY,PFILE2
      INQUIRE (FILE=PFILE2,EXIST=FLAG3)
      IF (.NOT. FLAG3) GOTO 35

C      USER SELECTED A VALID PROCEDURE FILE
      CLOSE (PUNIT1)
      CALL PROPRO(ITR,PFILE2,PUNIT1)
      GOTO 100

      END
      SUBROUTINE PROPRO(ITR,PFILE,PUNIT)

C*****
C  THIS SUBROUTINE PROCESS THE PROCEDURE FILE.  IT DISPLAYS THE FILE ON THE
C  SCREEN 23 LINES AT A TIME
C  VARIABLES PASSED:
C
C  ITR    -  INTERACTIVE TERMINAL READ UNIT
C  PUNIT  -  UNIT # TO OPEN THE PROCEDURE FILE (PFILE) ON
C  PFILE  -  PROCEDURE FILE NAME TO DISPLAY
C*****

      CHARACTER*80  DLINE
      CHARACTER*7   PFILE
      CHARACTER*1   INP(1),FMFEED

      INTEGER       PUNIT,RC(1,3)

      LOGICAL       FLAG1,IFLAG

      RC(1,1)=23
      RC(1,2)=34
      RC(1,3)=1

C      CLEAR THE SCREEN AND OPEN THE PROCEDURE FILE PROCXX
      CALL CLEAR(7,0)

```

```

OPEN (PUNIT, FILE=PFILE, STATUS='OLD')

IFLAG=.FALSE.
FLAG1=.TRUE.
30  DISPLAY 22 LINES OF TEXT ON THE SCREEN
C    DO 10 I=1,22,1
      READ(PUNIT,'(A80)',END=15) DLINE
      IF (FLAG1) THEN
        FLAG1=.FALSE.
        CALL CLEAR(7,0)
      ENDIF
      CALL MENU DR(DLINE, I, 1, 2, 0, 1, 1)
10  CONTINUE
    GOTO 16

15  IFLAG=.TRUE.

16  IF (IFLAG) CALL MENU DR('END OF FILE REACHED',23,62,7,0,1,1)
    CALL MENU DR('SELECT OPTION (X OR C OR P) ==>',23,1,2,0,1,1)

C    INPUT USERS SELECTION
20  INP(1)=' '
    CALL MENU DR(RC, 1, 1, 1, INP, ITR)

C    USER SELECTED TO RETURN
    IF (INP(1) .EQ. 'X') THEN
      CLOSE (PUNIT)
      RETURN
    ENDIF

C    USER SELECTED TO PRINT THE PROCEDURE FILE
    IF (INP(1) .EQ. 'P') THEN
      CLOSE(PUNIT)
      OPEN(PUNIT, FILE=PFILE, STATUS='OLD')
      IF= 12
      FMFEED= CHAR(IF)
C    OPEN THE PRINTER ON UNIT 15
      OPEN(15, FILE='/DEV/PRT')
      READ(PUNIT,'(A80)',END=220) DLINE
200  CONTINUE
C    SEND FORM FEED
      WRITE(15,'(A1)') FMFEED
      DO 210 I=1,22
        READ(PUNIT,'(A80)',END=220) DLINE
        WRITE(15,'(A80)') DLINE
210  CONTINUE
      GO TO 200

C
220  CONTINUE
      WRITE(15,'(A1)') FMFEED
      CLOSE(PUNIT)
      CLOSE(15)
      RETURN
    ENDIF

C    USER SELECTED TO CONTINUE VIEWING THE PROCEDURE FILE
    IF (INP(1) .EQ. 'C') THEN
      IF (IFLAG) THEN
        IFLAG=.FALSE.
        CLOSE (PUNIT)

```

```

                                OPEN (PUNIT, FILE=PFILE, STATUS='OLD')
                                ENDIF

                                GOTO 30
                                ENDIF

                                GOTO 20
                                END
                                SUBROUTINE PDEL(ITR, PUNIT1, PUNIT2, PFILE1, SMUNIT, SMFILE)

C*****
C THIS SUBROUTINE DELETES PROCEDURE FILES FROM THE PROCEDURE DIRECTORY FILE
C ONLY
C   VARIABLES PASSED:
C
C   ITR      -   INTERACTIVE TERMINAL READ UNIT
C   PUNIT1   -   UNIT # TO OPEN LISTING FILE (PFILE1) ON
C   PUNIT2   -   UNIT # TO OPEN PROCEDURE FILE ON
C   PFILE1   -   PROCEDURE LISTING FILE NAME
C   SMUNIT   -   UNIT # TO OPEN MENU FILE (SMFILE) ON
C   SMFILE   -   MENU FILE NAME
C*****

                                CHARACTER*70  OUT(19)
                                CHARACTER*40  PKEY
                                CHARACTER*18  CMD(1), INP
                                CHARACTER*7   PFILE1, PFILE2, EFILE, SMFILE

                                INTEGER        ITR, JJ, CNT, PUNIT1, PUNIT2, ISTART, IEND, ST(3), RC(20, 3)

                                LOGICAL        FLAG1, FLAG2, FLAG3

                                DATA ST/0, 0, 0/

C   DISPLAY THE MAIN MENU
                                CALL MENUSV(SMFILE, 148, RC, 20, SMUNIT)

C   CHECK TO SEE THAT THE PROCEDURE DATA BASE EXISTS
                                INQUIRE (FILE=PFILE1, EXIST=FLAG1)

C   NO PROCEDURE DIRECTORY FILE, DISPLAY A MESSAGE
                                IF (.NOT. FLAG1) THEN
C
C                                     CALL MESS(18, RC(20, 1), RC(20, 2), RC(20, 3), 7)
5                                     CMD(1)(1:18)=' '
C                                     CALL MENURD(RC, 20, 1, 1, CMD, ITR)
C                                     IF (CMD(1) .EQ. 'X') RETURN
C                                     GOTO 5
C                                     ENDIF

C   OPEN THE PROCEDURE DIRECTORY FILE
                                OPEN (PUNIT1, FILE=PFILE1, STATUS='OLD', ACCESS='DIRECT',
*   FORM='UNFORMATTED', RECL=48)

C   FIND OUT HOW MANY RECORDS ARE IN IT
15  READ(PUNIT1, REC=1) NREC
                                ISTART=-1
                                FLAG1=.TRUE.
                                FLAG2=.TRUE.
                                CNT=2

C   DISPLAY THE PROCEDURE FILES ON THE SCREEN
                                DO 20 I=2, NREC, 1

```

```

C      USER SELECTED TO CONTINUE VIEWING THE LIST
      IF (CMD(1) .EQ. 'C') THEN
          CALL PCOMP(PUNIT1,PFILE1)
          CALL MENUSV(SMFILE,148,RC,20,SMUNIT)
          GOTO 15
      ENDIF

```

```

C      DELETE AND COMPRESS THE PROCEDURE DIRECTORY FILE
      INP=CMD(1)
      CALL PROPDE(PUNIT1,PFILE1,INP,ISTART,IEND)
      GOTO 35

```

```

      END
      SUBROUTINE PROPDE(PUNIT1,PFILE1,L,ISTART,IEND)

```

```

C*****
C      THIS PROCEDURE DELETES THE PROCEDURE FILE NAME FROM THE DIRECTORY AND
C      COMPRESS THE FILE
C      VARIABLES PASSED:

```

```

C      PUNIT1 - UNIT # TO OPEN THE PROCEDURE DIRECTORY FILE ON
C      PFILE1 - PROCEDURE DIRECTORY FILE NAME
C      L      - LIST OF THE FILES TO BE DELETED
C      ISTART - STARTING FILE NUMBER
C      IEND   - ENDING FILE NUMBER
C*****

```

```

      CHARACTER*48 XOUT
      CHARACTER*18 L
      CHARACTER*8  FMT
      CHARACTER*7  PFILE1
      CHARACTER*1  C1
      INTEGER      PUNIT1,CNT,JS,ST,POS(8),IL,ISTART,IEND

```

```

      XOUT='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'

```

```

      FMT='(BN, IXX)'

```

```

      CNT=1

```

```

      ST=1

```

```

      DO 100 I=1,18,1

```

```

          IF (ST .EQ. 1) THEN

```

```

              I1=INDEX('0123456789 ',L(I:I))

```

```

              IF (I1 .EQ. 11) GOTO 100

```

```

              IF (I1 .EQ. 0) THEN

```

```

                  CALL MENU DR('INVALID',1,55,7,0,1,1)

```

```

                  CALL MENU DR(L,1,63,2,0,1,1)

```

```

                  C1=L(I:I)

```

```

                  CALL MENU DR(C1,1,62+I,7,0,1,1)

```

```

                  RETURN

```

```

              ENDIF

```

```

              JS=I

```

```

              ST=2

```

```

              GOTO 100

```

```

          ENDIF

```

```

      IF (ST .EQ. 2) THEN

```

```

          I1=INDEX('0123456789 ',L(I:I))

```

```

          IF (I1 .EQ. 0) THEN

```

```

              CALL MENU DR('INVALID',1,55,7,0,1,1)

```

```

              CALL MENU DR(L,1,63,2,0,1,1)

```

```

              C1=L(I:I)

```

```

CALL MENUADR(C1,1,62+I,7,0,1,1)
RETURN
ENDIF
IF (I1 .NE. 12) GOTO 100
IL=I-JS
IF (IL .GE. 10) THEN
50  WRITE(FMT(6:7),50) IL
    FORMAT(I2)
    ELSE
    FMT(7:7)=' '
    WRITE(FMT(6:6),55) IL
55  FORMAT(I1)
    ENDIF
READ(L(JS:I-1),FMT) POS(CNT)
CNT=CNT+1
ST=3
GOTO 100
ENDIF
IF (ST .EQ. 3) THEN
    I1=INDEX('0123456789 ',L(I:I))
    IF (I1 .EQ. 11) GOTO 100
    IF (I1 .EQ. 0) THEN
        CALL MENUADR('INVALID',1,55,7,0,1,1)
        CALL MENUADR(L,1,63,2,0,1,1)
        C1=L(I:I)
        CALL MENUADR(C1,1,62+I,7,0,1,1)
        RETURN
    ENDIF
    JS=I
    ST=2
    ENDIF
100  CONTINUE
IF (ST .EQ. 2) THEN
    IL=I-JS
    IF (IL .GE. 10) THEN
        WRITE(FMT(6:7),50) IL
        ELSE
        FMT(7:7)=' '
        WRITE(FMT(6:6),55) IL
        ENDIF
    READ(L(JS:18),FMT) POS(CNT)
    ENDIF
DO 150 I=1,CNT,1
    IF ((POS(I) .LT. ISTART).OR.(POS(I) .GT. IEND)) GOTO 150
    I1=6+POS(I)-ISTART
    CALL MENUADR('DELETED',I1,55,7,0,1,1)
    WRITE(PUNIT1,REC=POS(I)) XOUT
150  CONTINUE
RETURN
END
SUBROUTINE PCOMP(PUNIT,PFILE)

```

```

C*****
C  THIS SUBROUTINE COMPRESSES THE PROCEDURE LIST FILE
C  VARIABLES PASSED:
C
C  PUNIT - UNIT # PFILE IS OPEN ON

```

```

C      PFILE - NOT USED
C*****
      CHARACTER*48  LINEIN,XOUT
      CHARACTER*7   PFILE

      INTEGER       PUNIT,NREC,I

      XOUT='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'

C      FIND OUT HOW MANY RECORDS ARE IN THE FILE
      READ(PUNIT,REC=1) NREC
      I=2

C      UPDATE THE HEADER RECORD TO REFLECT THE COMPRESSION
100    IF (I .GT. NREC) THEN
           WRITE(PUNIT,REC=1) NREC
           RETURN
           ENDIF

C      INPUT THE PROCEDURE FILE NAME
      READ(PUNIT,REC=I) LINEIN

C      IF IT IS ALL X'S THEN MOVE THE FILE DOWN ONE
      IF (LINEIN .EQ. XOUT) THEN
           DO 200 J=I,NREC-1,1
               READ(PUNIT,REC=J+1) LINEIN
               WRITE(PUNIT,REC=J) LINEIN
           CONTINUE
           NREC=NREC-1
           GOTO 100
           ENDIF

      I=I+1
      GOTO 100

      END

```

Filename - ssdb.cor

\$bigcode

\$segment %ssdbseg

```
SUBROUTINE SSSIP(ITR,SUNIT1,SUNIT2,SUNIT3,SUNIT4,SFILE1,
*           SFILE2,SFILE3,SFILE4,SHFILE,SMUNIT,
*           SMFILE)
```

C THIS SUBROUTINE MANAGES THE SUBSTANCE-SOURCE DATA BASE

```
CHARACTER*1 OPT,CMD(1)
CHARACTER*7 SFILE1,SFILE2,SFILE3,SFILE4,SHFILE,SMFILE
CHARACTER*40 SKEY(2)
```

```
REAL        SDATA(14)
```

```
INTEGER     ITR,eFLAG,SUNIT1,SUNIT2,SUNIT3,SUNIT4,SMUNIT,
*           RC(2,3)
```

1 CALL MENUSV(SMFILE,170,RC,2,SMUNIT)

2 CMD(1)=' '

CALL MENURD(RC,2,1,1,CMD,ITR)

C CHECK FOR A VALID INPUT

IF (INDEX('12345X',CMD(1)) .EQ. 0) THEN

CALL MESS(11,RC(2,1),RC(2,2),RC(2,3),6)
GOTO 2

ENDIF

OPT=' '

IF (CMD(1) .EQ. '1') CALL SHELP(ITR,SUNIT1,SHFILE,SMUNIT,SMFILE)

IF (CMD(1) .EQ. '2') CALL SDEL(ITR,SUNIT1,SUNIT2,SUNIT3,SUNIT4,
* SFILE1,SFILE2,SFILE3,SFILE4,SMUNIT,
* SMFILE)

IF (CMD(1) .EQ. '3') CALL SADD(ITR,SUNIT1,SUNIT2,SUNIT3,SUNIT4,
* SFILE1,SFILE2,SFILE3,SFILE4,SMUNIT,
* SMFILE)

IF (CMD(1) .EQ. '4') CALL SMOD(ITR,SUNIT1,SUNIT2,SUNIT3,SUNIT4,
* SFILE1,SFILE2,SFILE3,SFILE4,SMUNIT,
* SMFILE)

IF (CMD(1) .EQ. '5') CALL SSEAR(EFLAG,ITR,SUNIT1,SUNIT2,SUNIT3,
* SUNIT4,OPT,SKEY,SDATA,SFILE1,SFILE2,
* SFILE3,SFILE4,SMUNIT,SMFILE)

IF (CMD(1) .EQ. 'X') RETURN

GOTO 1

END

SUBROUTINE SHELP(ITR,SUNIT,SHFILE,SMUNIT,SMFILE)

```
CHARACTER*80 TLINE,LINE
CHARACTER*7 SHFILE,SMFILE
CHARACTER*1 CMD(1)
```

```
INTEGER     ITR,SUNIT,SMUNIT,RC(3,3)
```

```
LOGICAL     FLAG
```

1 CALL MENUSV(SMFILE,175,RC,3,SMUNIT)

2 CMD(1)=' '

CALL MENURD(RC,3,1,1,CMD,ITR)

C CHECK FOR A VALID INPUT

```

IF (INDEX('1234X',CMD(1)) .EQ. 0) THEN
    CALL MESS(11, RC(2, 1), RC(2, 2), RC(2, 3), 6)
    GOTO 2
ENDIF

IF (CMD(1) .EQ. 'X') RETURN
INQUIRE (FILE=SHFILE, EXIST=FLAG)
IF (.NOT. FLAG) THEN
    CALL MESS(15, RC(2, 1), RC(2, 2), RC(2, 3), 6)
    GOTO 2
ENDIF

CALL CLEAR(7, 0)
OPEN (SUNIT, FILE=SHFILE, STATUS='OLD')
TLINE(1:80)='*X*'
TLINE(2:2)=CMD(1)
5 READ(SUNIT, '(A80)') LINE
IF (TLINE(1:3) .EQ. LINE(1:3)) THEN
    READ(LINE(4:5), '(I2)') IG
    J=0
    DO 20 I=1, IG, 1
        READ(SUNIT, '(A80)') LINE
        J=J+1
        CALL MENU DR(LINE, J, 1, 2, 0, 1, 1)
        IF (MOD(I, 22) .EQ. 0) THEN
            IF (I .EQ. IG) THEN
                CALL MESS(19, RC(3, 1), RC(3, 2), RC(3, 3), 7)
                READ(ITR, '(A1)') CMD(1)
                CLOSE (SUNIT)
                GOTO 1
            ELSE
                CALL MESS(16, RC(3, 1), RC(3, 2), RC(3, 3), 6)
                READ(ITR, '(A1)') CMD(1)
                CALL CLEAR(7, 0)
                J=0
            ENDIF
        ENDIF
    ENDIF
20 CONTINUE
    CALL MESS(19, RC(3, 1), RC(3, 2), RC(3, 3), 6)
    READ(ITR, '(A1)') CMD(1)
    CLOSE (SUNIT)
    GOTO 1
ENDIF

GOTO 5
END
SUBROUTINE SADD(ITR, SUNIT1, SUNIT2, SUNIT3, SUNIT4,
*          SFILE1, SFILE2, SFILE3, SFILE4, SMUNIT,
*          SMFILE)
C THIS SUBROUTINE ADDS ELEMENTS TO THE SUBSTANCE-SOURCE DATA BASE

CHARACTER*40 T1(2), CINIT1, CINIT2, TEMP0
CHARACTER*8 T2(11)
CHARACTER*7 SFILE1, SFILE2, SFILE3, SFILE4, SMFILE
CHARACTER*1 INP(1)

INTEGER ISET, ITR, I, J, SUNIT1, SUNIT2, SUNIT3, IST, JST,
* SUNIT4, RC(13, 3), ST(3), ITT, HD1, HD2, HD4

INTEGER SINIT1(10), HD3(10)

```

```

REAL          S2(10), S3(4), S4(11)

LOGICAL       FLAG, FLAG1, FLAG2, FLAG3, FLAG4

DATA ST /0,0,0/

ITT=-1

2000 DO 5 I=1,11,1
      T2(I)(1:8)=' '
5     CONTINUE
      T1(1)(1:40)=' '
      T1(2)(1:40)=' '

C     DISPLAY THE MAIN MENU
      CALL MENUSV(SMFILE, 180, RC, 13, SMUNIT)

C     READ IN THE SUBSTANCE AND THE SOURCE FROM THE MENU
      IST=1
415   CALL MENURD(RC, 13, IST, 2, T1, ITR)

C     CHECK TO SEE IF THE SUBSTANCE OR SOURCE IS BLANK
C     CAN NOT GO ON IF EITHER SUBSTANCE OR SOURCE IS BLANK
      IF ((T1(1) .EQ. ' ').OR. (T1(2) .EQ. ' ')) THEN
        IF (T1(2) .EQ. ' ') IST=2
        IF (T1(1) .EQ. ' ') IST=1
        GOTO 415
      ENDIF

      IF ((T1(1) .EQ. '?').OR. (T1(2) .EQ. '?')) THEN
        IF (T1(2) .EQ. '?') IST=2
        IF (T1(1) .EQ. '?') IST=1
        GOTO 415
      ENDIF

C     CHECK TO SEE IF THE SUBSTANCE IS IN THE DATA BASE   IF SO DISPLAY
C     THE GMW, 10, 30 AND 60 MIN PEL AND DO NOT ALLOW THEM TO BE MODIFIED
C
C     ISET=2 ==> THAT THE USER ENTERED SUBSTANCE DATA AND SHOULD BE ALLOWED
C     TO CHANGE IT.
C     ISET=3 ==> THE SUBSTANCE DATA CAME FROM THE DATA BASE AND THE USER
C     SHOULD NOT BE ALLOWED TO CHANGE IT.
      ISET=2
      INQUIRE (FILE=SFIL1, EXIST=FLAG)
      IF (FLAG) THEN
        OPEN (SUNIT1, FILE=SFIL1, STATUS='OLD', ACCESS='DIRECT',
          *      FORM='UNFORMATTED', RECL=120)
        READ(SUNIT1, REC=1) HD1
        DO 450 I=2, HD1, 1
          READ(SUNIT1, REC=I) CINIT1, SINIT1, S2
          IF (CINIT1 .EQ. T1(1)) THEN
            DO 460 J=1, 5, 1
              WRITE(T2(J+2)(1:8), '(F8.4)') S2(J)
460          CONTINUE
              ST(3)=0
              CALL MENUWR(RC, 13, 3, 7, T2, 0, 1, ST)
              CLOSE (SUNIT1)
              ISET=3
              JST=8

```

```

                                GOTO 320
                                ENDIF
450    CONTINUE
        CLOSE (SUNIT1)
        ENDIF

C      DISPLAY THE REST OF THE MENU INCLUDING GMW, 10, 30, 60 MIN
C      PELS IF NECESSARY.
        IST=3
420    CALL MENURD(RC, 13, IST, 7, T2, ITR)

        JST=8
320    CALL MENURD(RC, 13, JST, 11, T2, ITR)

C      SCANNING THE COMMAND INPUT ROW
200    INP(1)=' '
        CALL MENURD(RC, 13, 12, 12, INP, ITT)
        IF (INP(1) .EQ. ' ') THEN
            ST(3)=1
            CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)
            CALL MESS(4, RC(13, 1), RC(13, 2), RC(13, 3), 2)
            IF (ISET .EQ. 3) THEN
                JST=8
                GOTO 320
            ENDIF
            IF (ISET .EQ. 2) THEN
                IST=3
                GOTO 420
            ENDIF
            DO 210 J=1, 7, 1
                T2(J)(1:8)=' '
210        CONTINUE
                IST=1
                GOTO 415
            ENDIF
            IF (INP(1) .EQ. 'X') RETURN
            IF (INP(1) .EQ. 'R') GOTO 2000
            IF (INP(1) .NE. 'A') GOTO 200

C      CHECK THE REQUEST FOR ERRORS
        CALL MENUCK(T2, S4, 11, ' (F4.4)', IERR)
        IF (IERR .NE. 0) THEN
            INP(1)=' '
            ST(3)=1
            CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)
            CALL MESS(1, RC(13, 1), RC(13, 2), RC(13, 3), 6)
            IST=IERR
            IF (IERR .LE. 7) GOTO 420
            GOTO 320
        ENDIF

C      THE GMW, 10, 30, 60 MIN PELS AND THE SOURCE STRENGTH ARE NOT
C      ALLOWED TO BE ZERO.
        DO 180 I=3, 6, 1
            IF (S4(I) .LE. 0.0) THEN

```

```

                                IST=I
                                INP(1)=' '
                                ST(3)=1
                                CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)
                                CALL MESS(1, RC(13, 1), RC(13, 2), RC(13, 3), 6)
                                GOTO 420
                                ENDIF

180  CONTINUE
    IF (S4(8) .LE. 0.0) THEN
        JST=8
        INP(1)=' '
        ST(3)=1
        CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)
        CALL MESS(1, RC(13, 1), RC(13, 2), RC(13, 3), 6)
        GOTO 320
        ENDIF

C   LOAD THE S2(1-10) AND S3(1-4) ARRAYS
    S3(1)=S4(8)
    S3(2)=S4(9)
    S3(3)=S4(10)
    S3(4)=S4(11)
    DO 181 I=1, 10, 1
        S2(I)=0.0
181  CONTINUE
    DO 182 I=1, 5, 1
        S2(I)=S4(I+2)
182  CONTINUE

C   IF NO DATA BASE EXISTS THEN INITIALIZE IT
    INQUIRE (FILE=SFILE1, EXIST=FLAG)
    IF (.NOT. FLAG) THEN
        OPEN (SUNIT1, FILE=SFILE1, STATUS='NEW',
            *   ACCESS='DIRECT', FORM='UNFORMATTED', RECL=120)
        OPEN (SUNIT2, FILE=SFILE2, STATUS='NEW',
            *   ACCESS='DIRECT', FORM='UNFORMATTED', RECL=40)
        OPEN (SUNIT3, FILE=SFILE3, STATUS='NEW',
            *   ACCESS='DIRECT', FORM='UNFORMATTED', RECL=40)
        OPEN (SUNIT4, FILE=SFILE4, STATUS='NEW',
            *   ACCESS='DIRECT', FORM='UNFORMATTED', RECL=16)

C   INITIALIZE THE HEADERS
        HD1=1
        WRITE(SUNIT1, REC=1) HD1
        WRITE(SUNIT2, REC=1) HD1
        WRITE(SUNIT3, REC=1) HD1
        WRITE(SUNIT4, REC=1) HD1
        WRITE(SUNIT1, REC=2) HD1
        WRITE(SUNIT2, REC=2) HD1
        WRITE(SUNIT3, REC=2) HD1
        WRITE(SUNIT4, REC=2) HD1
        CLOSE(UNIT=SUNIT1)
        CLOSE(UNIT=SUNIT2)
        CLOSE(UNIT=SUNIT3)
        CLOSE(UNIT=SUNIT4)
        ENDIF

    OPEN (SUNIT1, FILE=SFILE1, STATUS='OLD', RECL=120, ACCESS='DIRECT',

```

```

*      FORM='UNFORMATTED')
OPEN (SUNIT2,FILE=SFILE2,STATUS='OLD',RECL=40,ACCESS='DIRECT',
*      FORM='UNFORMATTED')
OPEN (SUNIT3,FILE=SFILE3,STATUS='OLD',RECL=40,ACCESS='DIRECT',
*      FORM='UNFORMATTED')
OPEN (SUNIT4,FILE=SFILE4,STATUS='OLD',RECL=16,ACCESS='DIRECT',
*      FORM='UNFORMATTED')

```

C CHECK FOR SUBSTANCE IN THE SUBSTANCE DATA BASE

```
READ(SUNIT1,REC=1) HD1
```

```
DO 35 I=2,HD1,1
```

```
READ(SUNIT1,REC=I) CINIT1,SINIT1
```

```
IF (CINIT1.EQ. T1(1)) THEN
```

```
READ(SUNIT2,REC=1) HD2
```

```
DO 42 J=2,HD2,1
```

```
READ(SUNIT2,REC=J) CINIT2
```

```
IF (CINIT2.EQ. T1(2)) THEN
```

```
SEE IF THE ENTRY ALREADY EXISTS
```

```
DO 26 K=1,9,1
```

```
IF (SINIT1(K).EQ. J) THEN
```

```
CALL MESS(2,RC(13,1),RC(13,2),RC(13,3),6)
```

```
GOTO 1000
```

```
ENDIF
```

26

```
CONTINUE
```

```
IF (SINIT1(10).NE. 0) THEN
```

```
READ(SUNIT1,REC=SINIT1(10)) CINIT1,
SINIT1
```

```
GOTO 25
```

```
ENDIF
```

```
READ(SUNIT1,REC=I) CINIT1,SINIT1
```

```
READ(SUNIT4,REC=1) HD4
```

```
HD4=HD4+1
```

```
WRITE(SUNIT4,REC=1) HD4
```

```
III=HD4+1
```

```
WRITE(SUNIT4,REC=HD4) S3
```

```
WRITE(SUNIT4,REC=III) S3
```

28

```
DO 27 KK=1,9,1
```

```
IF (SINIT1(KK).EQ. 0) THEN
```

```
SINIT1(KK)=J
```

```
III=I+1
```

```
WRITE(SUNIT1,REC=I) CINIT1,SINIT1,S2
```

```
WRITE(SUNIT1,REC=III) CINIT1
```

```
READ(SUNIT3,REC=1) HD3
```

```
HD3(KK)=HD4
```

```
WRITE(SUNIT3,REC=1) HD3
```

```
WRITE(SUNIT3,REC=III) HD3
```

```
CALL MESS(3,RC(13,1),RC(13,2),RC(13,3),6)
```

```
GOTO 1000
```

```
ENDIF
```

27

```
CONTINUE
```

```
IF (SINIT1(10).EQ. 0) THEN
```

```
HD1=HD1+1
```

```
SINIT1(10)=HD1
```

```
WRITE(SUNIT1,REC=I) CINIT1,SINIT1,S2
```

```

29      WRITE(SUNIT1,REC=1) HD1
        DO 29 K=2,10,1
          SINIT1(K)=0
          HD3(K)=0
        CONTINUE
        SINIT1(1)=J
        HD3(1)=HD4
        III=HD1+1
        WRITE(SUNIT1,REC=HD1) CINIT1,SINIT1,S2
        WRITE(SUNIT3,REC=HD1) HD3
        WRITE(SUNIT1,REC=III) CINIT1
        WRITE(SUNIT3,REC=III) HD3
        CALL MESS(3,RC(13,1),RC(13,2),RC(13,3),6)
        GOTO 1000
                                ENDIF
        I=SINIT1(10)
        READ(SUNIT1,REC=I) CINIT1,SINIT1
        GOTO 28
                                ENDIF
42      CONTINUE
C      ADD NEW SOURCE TO THE SOURCE DATA BASE
        HD2=HD2+1
        WRITE(SUNIT2,REC=1) HD2
        III=HD2+1
        WRITE(SUNIT2,REC=HD2) T1(2)
        WRITE(SUNIT2,REC=III) T1(2)
        READ(SUNIT4,REC=1) HD4
        HD4=HD4+1
        WRITE(SUNIT4,REC=1) HD4
        III=HD4+1
        WRITE(SUNIT4,REC=HD4) S3
        WRITE(SUNIT4,REC=III) S3
46      DO 45 JJ=1,9,1
        IF (SINIT1(JJ) .EQ. 0) THEN
          SINIT1(JJ)=HD2
          WRITE(SUNIT1,REC=1) CINIT1,SINIT1,S2
          READ(SUNIT3,REC=1) HD3
          HD3(JJ)=HD4
          WRITE(SUNIT3,REC=1) HD3
          CALL MESS(3,RC(13,1),RC(13,2),RC(13,3),6)
          GOTO 1000
                                ENDIF
45      CONTINUE
        IF (SINIT1(10) .EQ. 0) THEN
          HD1=HD1+1
          SINIT1(10)=HD1
          WRITE(SUNIT1,REC=1) CINIT1,SINIT1,S2
          WRITE(SUNIT1,REC=1) HD1
          DO 47 K=2,10,1
            SINIT1(K)=0
            HD3(K)=0
          CONTINUE
          SINIT1(1)=HD2
          HD3(1)=HD4
          III=HD1+1
          WRITE(SUNIT1,REC=HD1) CINIT1,SINIT1,S2
          WRITE(SUNIT3,REC=HD1) HD3
          WRITE(SUNIT1,REC=III) CINIT1
          WRITE(SUNIT3,REC=III) S3
47

```

CALL MESS(3, RC(13, 1), RC(13, 2), RC(13, 3), 6)
GOTO 1000

ENDIF

I=SINIT1(10)
READ(SUNIT1, REC=I) CINIT1, SINIT1
GOTO 46

ENDIF

35 CONTINUE

C A NEW ADDITION IS TO BE MADE TO SUB.DB
C THE SOURCE IS IN SOUR.DB BUT THE SUBSTANCE IS NEW IN SUB.DB

READ(SUNIT2, REC=1) HD2

DO 50 J=2, HD2, 1

READ(SUNIT2, REC=J) CINIT2

IF (CINIT2.EQ. T1(2)) THEN

DO 55 I=2, 10, 1

SINIT1(I)=0

HD3(I)=0

55

CONTINUE

HD1=HD1+1

SINIT1(1)=J

READ(SUNIT4, REC=1) HD4

WRITE(SUNIT1, REC=1) HD1

III=HD1+1

WRITE(SUNIT1, REC=HD1) T1(1), SINIT1, S2

WRITE(SUNIT1, REC=III) T1(1)

HD4=HD4+1

JJJ=HD4+1

WRITE(SUNIT4, REC=HD4) S3

WRITE(SUNIT4, REC=JJJ) S3

WRITE(SUNIT4, REC=1) HD4

HD3(1)=HD4

WRITE(SUNIT3, REC=HD1) HD3

WRITE(SUNIT3, REC=III) HD3

CALL MESS(3, RC(13, 1), RC(13, 2), RC(13, 3), 6)

GOTO 1000

ENDIF

50 CONTINUE

C A NEW ADDITION IS TO BE MADE TO BOTH SUB.DB AND SOUR.DB
C THE SUBSTANCE AND THE SOURCE ARE NOT IN THE DATA BASE.

READ(SUNIT4, REC=1) HD4

DO 60 I=2, 10, 1

SINIT1(I)=0

HD3(I)=0

60

CONTINUE

HD1=HD1+1

HD2=HD2+1

SINIT1(1)=HD2

WRITE(SUNIT1, REC=1) HD1

WRITE(SUNIT2, REC=1) HD2

WRITE(SUNIT1, REC=HD1) T1(1), SINIT1, S2

WRITE(SUNIT2, REC=HD2) T1(2)

```

      III= HD1+1
      JJJ= HD2+1
      WRITE(SUNIT1,REC=III) T1(1)
      WRITE(SUNIT2,REC=JJJ) T1(2)
      HD4=HD4+1
      WRITE(SUNIT4,REC=HD4) S3
      WRITE(SUNIT4,REC=1) HD4
      KKK= HD4+1
      WRITE(SUNIT4,REC=KKK) S3
      HD3(1)=HD4
      WRITE(SUNIT3,REC=HD1) HD3
      WRITE(SUNIT3,REC=III) HD3
      CALL MESS(3, RC(13,1), RC(13,2), RC(13,3), 6)
1000 CLOSE (SUNIT1)
      CLOSE (SUNIT2)
      CLOSE (SUNIT3)
      CLOSE (SUNIT4)
      ISET=1
      GOTO 200

      END
      SUBROUTINE SSEAR(EFLAG, ITR, SUNIT1, SUNIT2, SUNIT3, SUNIT4, OPT, SKEY,
*          SDATA, SFILE1, SFILE2, SFILE3, SFILE4, SMUNIT, SMFILE)
C THIS SUBROUTINE SEARCHES FOR ELEMENTS FROM THE SOURCE-SUBSTANCE DATA BASE

      CHARACTER*1 INP(1), OPT, OPT1
      CHARACTER*40 CINIT1, CINIT2, SKEY(1), TC(1)
      CHARACTER*8 TEMP(8)
      CHARACTER*7 SFILE1, SFILE2, SFILE3, SFILE4, SMFILE

      INTEGER      ITR, I, SUNIT1, SUNIT2, SUNIT3, SUNIT4, SMUNIT,
*          RC(4,3), ST(3), RCC(7,3), EFLAG, HD

      INTEGER      SINIT1(10), HD3(10)

      REAL          SDATA(1), S2(10), S3(4)

      LOGICAL       FLAG1, FLAG2, FLAG3, FLAG4

      DATA ST /0,0,1/

      ITT=-1
      OPT1=' '

      IF (OPT .EQ. '&') GOTO 20

100  SKEY(1)(1:40)=' '
      SKEY(2)(1:40)=' '
105  CALL MENUSV(SMFILE, 181, RC, 4, SMUNIT)

C    IF ALREADY BEEN IN THE MENU ALLOW THE USER TO QUIT
      IF (OPT1 .EQ. '*') THEN
          ST(3)=0
          CALL MENUWR(RC, 4, 1, 2, SKEY, 0, 1, ST)
          GOTO 15
      ENDIF

      IST=1
5    CALL MENURD(RC, 4, IST, 2, SKEY, ITR)

```

```

C      CHECK THE REQUEST FOR ERRORS
      IF ((SKEY(1) .EQ. ' ') .OR. (SKEY(2) .EQ. ' ')) THEN
          IF (SKEY(2) .EQ. ' ') IST=2
          IF (SKEY(1) .EQ. ' ') IST=1
          GOTO 5
      ENDIF

C      SCANNING THE COMMAND INPUT ROW
15     INP(1)=' '
      CALL MENURD(RC,4,3,3,INP,ITT)
      IF (INP(1) .EQ. ' ') THEN
          ST(3)=1
          CALL MENUWR(RC,4,3,3,INP,0,1,ST)
          CALL MESS(4,RC(4,1),RC(4,2),RC(4,3),1)
          IST=1
          GOTO 5
      ENDIF
      IF (INP(1) .EQ. 'X') RETURN
      IF (INP(1) .NE. 'S') GOTO 15

C      ALLOW THE QUESTION MARK TO BE ANSWERED
      IF ((SKEY(1) .EQ. '?') .OR. (SKEY(2) .EQ. '?')) THEN
          INQUIRE (FILE=SFILE1,EXIST=FLAG1)
          INQUIRE (FILE=SFILE2,EXIST=FLAG2)
          IF ((.NOT.FLAG1) .OR. (.NOT.FLAG2)) THEN
              CALL MESS(18,RC(4,1),RC(4,2),RC(4,3),7)
              GOTO 15
          ENDIF
          IF (SKEY(1) .EQ. '?') THEN
              CALL SBQST(SKEY,SUNIT1,SUNIT2,SFILE1,SFILE2,SMUNIT,SMFILE,
*              ITR,EFLAG)
              IF (EFLAG .NE. 0) THEN
                  ST(3)=1
                  CALL MENUWR(RC,4,3,3,INP,0,1,ST)
                  CALL MESS(5,RC(4,1),RC(4,2),RC(4,3),1)
                  IST=2
                  GOTO 5
              ENDIF
          ENDIF
          IF (SKEY(2) .EQ. '?') THEN
              CALL SRCQST(SKEY,SUNIT1,SUNIT2,SFILE1,SFILE2,SMUNIT,SMFILE,
*              ITR,EFLAG)
              IF (EFLAG .NE. 0) THEN
                  ST(3)=1
                  CALL MENUWR(RC,4,3,3,INP,0,1,ST)
                  CALL MESS(5,RC(4,1),RC(4,2),RC(4,3),1)
                  IST=1
                  GOTO 5
              ENDIF
          ENDIF
      ENDIF

      OPT1='*'
      GOTO 105
  ENDIF

C      CHECK THAT THE DATABASES EXISTS
20     INQUIRE (FILE=SFILE1,EXIST=FLAG1)
      INQUIRE (FILE=SFILE2,EXIST=FLAG2)
      INQUIRE (FILE=SFILE4,EXIST=FLAG4)
      INQUIRE (FILE=SFILE3,EXIST=FLAG3)
      IF ((.NOT. FLAG1) .OR. (.NOT. FLAG2) .OR.

```

```

*      (.NOT. FLAG3).OR. (.NOT. FLAG4)) THEN
          IF (OPT .EQ. '&') THEN
              EFLAG=10
              RETURN
          ENDIF
          CALL MESS(5, RC(4, 1), RC(4, 2), RC(4, 3), 7)
              GOTO 15
          ENDIF

C      OPEN THE DATABASES
      OPEN (SUNIT1, FILE=SFILE1, STATUS='OLD', RECL=120, ACCESS='DIRECT',
*        FORM='UNFORMATTED')
      OPEN (SUNIT2, FILE=SFILE2, STATUS='OLD', RECL=40, ACCESS='DIRECT',
*        FORM='UNFORMATTED')
      OPEN (SUNIT3, FILE=SFILE3, STATUS='OLD', RECL=40, ACCESS='DIRECT',
*        FORM='UNFORMATTED')
      OPEN (SUNIT4, FILE=SFILE4, STATUS='OLD', RECL=16, ACCESS='DIRECT',
*        FORM='UNFORMATTED')

      FLAG1=.FALSE.
      FLAG2=.FALSE.
      FLAG4=.FALSE.
      READ(SUNIT1, REC=1) HD
      DO 25 I=2, HD, 1
          READ(SUNIT1, REC=I) CINIT1, SINIT1, S2
          IF ((SKEY(1) .EQ. CINIT1).OR. (SKEY(1) .EQ. '*')) THEN
              READ(SUNIT3, REC=I) HD3
              FLAG3=.FALSE.
              ICNT=12
              DO 30 II=1, 9, 1
                  IF (SINIT1(II) .EQ. 0) GOTO 30
                  FLAG1=.TRUE.
                  DO 80 IJ=1, 5, 1
                      SDATA(IJ)=S2(IJ)
80              CONTINUE
                  READ(SUNIT2, REC=SINIT1(II)) CINIT2
                  IF ((CINIT2 .EQ. SKEY(2)).OR.
*                    (SKEY(2) .EQ. '*')) THEN

                      READ(SUNIT4, REC=HD3(II)) S3
                      DO 81 IJ=1, 4, 1
                          SDATA(10+IJ)=S3(IJ)
81              CONTINUE
                  IF (OPT .EQ. '&') THEN
                      CLOSE (SUNIT1)
                      CLOSE (SUNIT2)
                      CLOSE (SUNIT3)
                      CLOSE (SUNIT4)
                      RETURN
                  ENDIF

          FLAG2=.TRUE.
C      FLAG3 IS A SWITCH USED TO DECIDE WHETHER TO WRITE HEADER
C      FLAG3 = .FALSE. ==> WRITE HEADER
C      FLAG3 = .TRUE. ==> DO NOT WRITE THE HEADER
          IF (.NOT. FLAG3) THEN
              FLAG3=.TRUE.
              CALL MENU5V(SMFILE, 184, RCC, 7, SMUNIT)
              ST(3)=0
              TC(1)=CINIT1

```

```

CALL MENUWR(RCC,7,1,1,TC,0,1,ST)
DO 40 K=1,5,1
  WRITE(TEMP(K+1)(1:8),'(F8.3)') S2(K)
40  CONTINUE
  ST(3)=0
  CALL MENUWR(RCC,7,2,6,TEMP,0,1,ST)
  ENDIF

DO 65 K=1,4,1
  WRITE(TEMP(K)(1:8),'(F8.1)') S3(K)
65  CONTINUE
  CALL MENUWR(CINIT2,ICNT,1,2,0,1,1)
  CALL MENUWR(TEMP(1)(1:8),ICNT,41,2,0,1,1)
  CALL MENUWR(TEMP(2)(1:8),ICNT,51,2,0,1,1)
  CALL MENUWR(TEMP(3)(1:8),ICNT,61,2,0,1,1)
  CALL MENUWR(TEMP(4)(1:8),ICNT,71,2,0,1,1)
  ICNT=ICNT+1

  ENDIF

30  CONTINUE

C    FLAG2 IS IS TO TELL IF ANYTHING WAS WRITTEN TO THE SCREEN
C    FLAG2 = .TRUE. ==> SOMETHING IS ON THE SCREEN
C    FLAG2 = .FALSE. ==> NOTHING WAS WRITTEN ON THE SCREEN
50  IF (FLAG2) THEN
    INP(1)=' '
    CALL MENUWR(RCC,7,7,7,INP,ITT)
    IF (INP(1).EQ.'X') THEN
      CLOSE (SUNIT1)
      CLOSE (SUNIT2)
      CLOSE (SUNIT3)
      CLOSE (SUNIT4)
      OPT1='*'
      GOTO 100
    ENDIF
    IF (INP(1).EQ.'C') THEN
      FLAG1=.FALSE.
      FLAG2=.FALSE.
      FLAG4=.TRUE.
      GOTO 25
    ENDIF
    GOTO 50
  ENDIF

25  CONTINUE
  ENDIF

  IF (((FLAG1).AND.(FLAG2)).OR.(FLAG4)) THEN
    CLOSE (SUNIT1)
    CLOSE (SUNIT2)
    CLOSE (SUNIT3)
    CLOSE (SUNIT4)
    OPT1='*'
    GOTO 100
  ENDIF

  IF (.NOT. FLAG1) THEN
    IF (OPT.EQ.'&') THEN
      EFLAG=11
      CLOSE (SUNIT1)
      CLOSE (SUNIT2)

```

```

CLOSE (SUNIT3)
CLOSE (SUNIT4)
RETURN
ENDIF

IST=1
GOTO 35
ENDIF
IF (.NOT. FLAG2) THEN
  IF (OPT .EQ. '&') THEN
    EFLAG=12
    CLOSE (SUNIT1)
    CLOSE (SUNIT2)
    CLOSE (SUNIT3)
    CLOSE (SUNIT4)
    RETURN
  ENDIF

  IST=2
ENDIF

35 INP(1)=' '
   ST(3)=1
   CALL MENUWR(RC, 4, 3, 3, INP, 0, 1, ST)
   CALL MESS(5, RC(4, 1), RC(4, 2), RC(4, 3), 7)
   CLOSE (SUNIT1)
   CLOSE (SUNIT2)
   CLOSE (SUNIT3)
   CLOSE (SUNIT4)
   GOTO 5

END
SUBROUTINE SDEL(ITR, SUNIT1, SUNIT2, SUNIT3, SUNIT4, SFILE1, SFILE2,
*          SFILE3, SFILE4, SMUNIT, SMFILE)
C THIS SUBROUTINE DELETES ELEMENTS FROM THE SOURCE-SUBSTANCE DATA BASE

CHARACTER*1 INP(1), OPT
CHARACTER*40 CINIT1, CINIT2, DKEY(2), TC(1)
CHARACTER*8 TEMP(10)
CHARACTER*7 SFILE1, SFILE2, SFILE3, SFILE4, SMFILE

INTEGER ITR, RC(4, 3), I, SUNIT1, SUNIT2, SUNIT3, SUNIT4, ST(3),
* RCC(7, 3), SMUNIT, EFLAG

INTEGER SINIT1(10), HD3(10), HD, SAV(10), SAV1(10)

REAL S2(10), S3(4)

LOGICAL FLAG1, FLAG2, FLAG3, FLAG4

DATA ST/0, 0, 0/

ITT=-1
OPT=' '
100 DKEY(1)(1:40)=' '
    DKEY(2)(1:40)=' '
105 CALL MENUSV(SMFILE, 185, RC, 4, SMUNIT)

IF (OPT .EQ. '&') THEN
  ST(3)=0
  CALL MENUWR(RC, 4, 1, 2, DKEY, 0, 1, ST)
  GOTO 15
ENDIF

```

```

      IST=1
5     CALL MENURD(RC,4,IST,2,DKEY,ITR)

C     CHECK THE REQUEST FOR ERRORS
      IF ((DKEY(1) .EQ. ' ') .OR. (DKEY(2) .EQ. ' ')) THEN
          IF (DKEY(2) .EQ. ' ') IST=2
          IF (DKEY(1) .EQ. ' ') IST=1
          GOTO 5
      ENDIF

C     SCANNING THE COMMAND INPUT ROW
15    INP(1)=' '
      CALL MENURD(RC,4,3,3,INP,ITT)
      IF (INP(1) .EQ. ' ') THEN
          ST(3)=1
          CALL MENUWR(RC,4,3,3,INP,0,1,ST)
          CALL MESS(4,RC(4,1),RC(4,2),RC(4,3),1)
          IST=1
          GOTO 5
      ENDIF
      IF (INP(1) .EQ. 'X') RETURN
      IF (INP(1) .NE. 'D') GOTO 15

C     ALLOW THE QUESTION MARK TO BE ANSWERED
      IF ((DKEY(1) .EQ. '?') .OR. (DKEY(2) .EQ. '?')) THEN
          INQUIRE (FILE=SFILE1,EXIST=FLAG1)
          INQUIRE (FILE=SFILE2,EXIST=FLAG2)
          IF ((.NOT.FLAG1) .OR. (.NOT.FLAG2)) THEN
              CALL MESS(18,RC(4,1),RC(4,2),RC(4,3),7)
              GOTO 15
          ENDIF

          IF (DKEY(1) .EQ. '?') THEN
              CALL SBQST(DKEY,SUNIT1,SUNIT2,SFILE1,SFILE2,SMUNIT,SMFILE,
*              ITR,EFLAG)
              IF (EFLAG .NE. 0) THEN
                  ST(3)=1
                  CALL MENUWR(RC,4,3,3,INP,0,1,ST)
                  CALL MESS(5,RC(4,1),RC(4,2),RC(4,3),1)
                  IST=2
                  GOTO 5
              ENDIF
          ENDIF

          IF (DKEY(2) .EQ. '?') THEN
              CALL SRCQST(DKEY,SUNIT1,SUNIT2,SFILE1,SFILE2,SMUNIT,SMFILE,
*              ITR,EFLAG)
              IF (EFLAG .NE. 0) THEN
                  ST(3)=1
                  CALL MENUWR(RC,4,3,3,INP,0,1,ST)
                  CALL MESS(5,RC(4,1),RC(4,2),RC(4,3),1)
                  IST=1
                  GOTO 5
              ENDIF
          ENDIF

          OPT='&'
          GOTO 105
      ENDIF

C     CHECK THAT THE DATA BASES EXIST
      INQUIRE (FILE=SFILE1,EXIST=FLAG1)

```

```

INQUIRE (FILE=SFILE2,EXIST=FLAG2)
INQUIRE (FILE=SFILE4,EXIST=FLAG4)
INQUIRE (FILE=SFILE3,EXIST=FLAG3)
IF ((.NOT. FLAG1).OR.(.NOT. FLAG2).OR.
*   (.NOT. FLAG3).OR.(.NOT. FLAG4)) THEN
    CALL MESS(5,RC(4,1),RC(4,2),RC94,3,7)
    GOTO 15
ENDIF

```

```

C   OPEN THE DATABASES
OPEN (SUNIT1,FILE=SFILE1,STATUS='OLD',RECL=120,ACCESS='DIRECT',
*   FORM='UNFORMATTED')
OPEN (SUNIT2,FILE=SFILE2,STATUS='OLD',RECL=40,ACCESS='DIRECT',
*   FORM='UNFORMATTED')
OPEN (SUNIT3,FILE=SFILE3,STATUS='OLD',RECL=40,ACCESS='DIRECT',
*   FORM='UNFORMATTED')
OPEN (SUNIT4,FILE=SFILE4,STATUS='OLD',RECL=16,ACCESS='DIRECT',
*   FORM='UNFORMATTED')

```

```

FLAG1=.FALSE.
FLAG2=.FALSE.
FLAG4=.FALSE.
READ(SUNIT1,REC=1) HD
DO 25 I=2,HD,1
    READ(SUNIT1,REC=I) CINIT1,SINIT1,S2
    IF ((DKEY(1) .EQ. CINIT1).OR.(DKEY(1) .EQ. '*')) THEN
        READ(SUNIT3,REC=I) HD3
        FLAG1=.TRUE.
        FLAG3=.FALSE.
        ICNT=12
        ICNTT=0
        DO 30 II=1,9,1
            IF (SINIT1(II) .EQ. 0) GOTO 30
            READ(SUNIT2,REC=SINIT1(II)) CINIT2
            IF ((CINIT2 .EQ. DKEY(2)).OR.
*           (DKEY(2) .EQ. '*')) THEN

```

```

        ICNTT=ICNTT+1
        SAV(ICNTT)=SINIT1(II)
        SAV1(ICNTT)=II
        FLAG2=.TRUE.
        READ(SUNIT4,REC=HD3(II)) S3

```

```

    IF (.NOT. FLAG3) THEN
        FLAG3=.TRUE.
        CALL MENUSV(SMFILE,184,RCC,7,SMUNIT)
        ST(3)=0
        TC(1)=CINIT1
        CALL MENUWR(RCC,7,1,1,TC,0,1,ST)
        DO 40 K=1,5,1
            WRITE(TEMP(K+1)(1:8),'(F8.3)') S2(K)
        CONTINUE
        ST(3)=0
        CALL MENUWR(RCC,7,2,6,TEMP,0,1,ST)
    ENDIF

```

40

```

    DO 65 K=1,4,1
        WRITE(TEMP(K)(1:8),'(F8.1)') S3(K)
    CONTINUE
    CALL MENUWR(CINIT2,ICNT,1,2,0,1,1)

```

65

```

CALL MENUUDR(TEMP(1)(1:8), ICNT, 41, 2, 0, 1, 1)
CALL MENUUDR(TEMP(2)(1:8), ICNT, 51, 2, 0, 1, 1)
CALL MENUUDR(TEMP(3)(1:8), ICNT, 61, 2, 0, 1, 1)
CALL MENUUDR(TEMP(4)(1:8), ICNT, 71, 2, 0, 1, 1)
ICNT=ICNT+1

                                                                ENDIF
30      CONTINUE

      IF (FLAG2) THEN
50      INP(1)=' '
      CALL MENURD(RCC, 7, 7, 7, INP, ITT)
      IF (INP(1) .EQ. 'X') THEN
                                                                CLOSE (SUNIT1)
                                                                CLOSE (SUNIT2)
                                                                CLOSE (SUNIT3)
                                                                CLOSE (SUNIT4)
                                                                OPT='&'
                                                                GOTO 100
                                                                ENDIF
      IF (INP(1) .EQ. 'C') THEN
        FLAG1=.FALSE.
        FLAG2=.FALSE.
        FLAG4=.TRUE.
        DO 80 JG=1, ICNT, 1
          SINIT1(SAV1(JG))=0
          HD3(SAV1(JG))=0
80      CONTINUE
        WRITE(SUNIT1, REC=1) CINIT1, SINIT1, S2
        WRITE(SUNIT3, REC=1) HD3
        GOTO 25
                                                                ENDIF
      GOTO 50
                                                                ENDIF

                                                                ENDIF
25      CONTINUE

      IF (((FLAG1).AND.(FLAG2)).OR.(FLAG4)) THEN
                                                                CLOSE (SUNIT1)
                                                                CLOSE (SUNIT2)
                                                                CLOSE (SUNIT3)
                                                                CLOSE (SUNIT4)
                                                                OPT='&'
                                                                GOTO 100
                                                                ENDIF

      IF (.NOT. FLAG2) IST=2
      IF (.NOT. FLAG1) IST=1

35      INP(1)=' '
      ST(3)=1
      CALL MENUWR(RC, 4, 3, 3, INP, 0, 1, ST)
      CALL MESS(5, RC(4, 1), RC(4, 2), RC(4, 3), 7)
      CLOSE (SUNIT1)
      CLOSE (SUNIT2)
      CLOSE (SUNIT3)
      CLOSE (SUNIT4)
      GOTO 5

      END

```

```

SUBROUTINE SMOD(ITR, SUNIT1, SUNIT2, SUNIT3, SUNIT4, SFILE1, SFILE2,
*          SFILE3, SFILE4, SMUNIT, SMFILE)
C THIS SUBROUTINE MODIFIES ELEMENTS FROM THE SOURCE-SUBSTANCE DATA BASE

CHARACTER*1 INP(1), OPT
CHARACTER*40 CINIT1, CINIT2, MKEY(2)
CHARACTER*8 TEMP(11)
CHARACTER*7 SFILE1, SFILE2, SFILE3, SFILE4, SMFILE

INTEGER      ITR, SUNIT1, SUNIT2, I, SUNIT3, SUNIT4, RC(13, 3), ST(3),
*            IERR, ITT, EFLAG
INTEGER      SINIT1(10), HD, HD3(10)

LOGICAL      FLAG1, FLAG2, FLAG3, FLAG4

REAL         S2(10), TDATA(11), S3(4)

DATA ST/0, 0, 0/

OPT=' '
ITT=-1
100 MKEY(1)(1:40)=' '
    MKEY(2)(1:40)=' '
105 CALL MENUSV(SMFILE, 186, RC, 13, SMUNIT)

IF (OPT .EQ. '&') THEN
    ST(3)=0
    CALL MENUWR(RC, 13, 1, 2, MKEY, 0, 1, ST)
    GOTO 15
ENDIF

C READ IN THE SUBSTANCE AND THE SOURCE FROM THE MENU
IST=1
5 CALL MENURD(RC, 13, IST, 2, MKEY, ITR)

C CHECK TO SEE IF THE SUBSTANCE OR THE SOURCE IS BLANK
C CAN NOT GO ON IF EITHER SUBSTANCE OR SOURCE IS BLANK
IF ((MKEY(1) .EQ. ' ') .OR. (MKEY(2) .EQ. ' ')) THEN
    IF (MKEY(2) .EQ. ' ') IST=2
    IF (MKEY(1) .EQ. ' ') IST=1
    GOTO 5
ENDIF

C SCANNING THE COMMAND INPUT ROW
15 INP(1)=' '
CALL MENURD(RC, 13, 12, 12, INP, ITT)
IF (INP(1) .EQ. ' ') THEN
    ST(3)=1
    CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)
    CALL MESS(4, RC(13, 1), RC(13, 2), RC(13, 3), 1)
    IST=1
    GOTO 5
ENDIF
IF (INP(1) .EQ. 'X') RETURN
IF (INP(1) .NE. 'M') GOTO 15

C ALLOW THE QUESTION MARK TO BE ANSWERED
IF ((MKEY(1) .EQ. '?') .OR. (MKEY(2) .EQ. '?')) THEN
    INQUIRE (FILE=SFILE1, EXIST=FLAG1)
    INQUIRE (FILE=SFILE2, EXIST=FLAG2)

```

```

IF ((.NOT.FLAG1).OR.(.NOT.FLAG2)) THEN
  CALL MESS(18, RC(13,1), RC(13,2), RC(13,3), 7)
  GOTO 15

```

ENDIF

```

IF (MKEY(1) .EQ. '?') THEN
  CALL SBQST(MKEY, SUNIT1, SUNIT2, SFILE1, SFILE2, SMUNIT, SMFILE,
    *      ITR, EFLAG)
  IF (EFLAG .NE. 0) THEN
    ST(3)=1
    CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)
    CALL MESS(5, RC(13,1), RC(13,2), RC(13,3), 1)
    IST=2
    GOTO 5
  ENDIF

```

ENDIF

```

IF (MKEY(2) .EQ. '?') THEN
  CALL SRCQST(MKEY, SUNIT1, SUNIT2, SFILE1, SFILE2, SMUNIT, SMFILE,
    *      ITR, EFLAG)
  IF (EFLAG .NE. 0) THEN
    ST(3)=1
    CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)
    CALL MESS(5, RC(13,1), RC(13,2), RC(13,3), 1)
    IST=1
    GOTO 5
  ENDIF

```

ENDIF

```

OPT='&'
GOTO 105

```

ENDIF

```

C CHECK TO SEE THAT ALL THE NECESSARY DATA BASES EXIST
  INQUIRE (FILE=SFILE1, EXIST=FLAG1)
  INQUIRE (FILE=SFILE2, EXIST=FLAG2)
  INQUIRE (FILE=SFILE3, EXIST=FLAG3)
  INQUIRE (FILE=SFILE4, EXIST=FLAG4)
  IF ((.NOT. FLAG1).OR.(.NOT. FLAG2).OR.
    *   (.NOT. FLAG3).OR.(.NOT. FLAG4)) THEN
    CALL MESS(18, RC(13,1), RC(13,2), RC(13,3), 7)
    GOTO 15
  ENDIF

```

```

C CHECK TO SEE IF THE SUBSTANCE IS IN THE DATA BASE
  OPEN (SUNIT1, FILE=SFILE1, STATUS='OLD', RECL=120, ACCESS='DIRECT',
    *   FORM='UNFORMATTED')
  READ(SUNIT1, REC=1) HD
  DO 20 I=2, HD, 1
    READ(SUNIT1, REC=I) CINIT1, SINIT1, S2
    IF (MKEY(1) .EQ. CINIT1) GOTO 25
20 CONTINUE
  IST=1
  INP(1)=' '
  ST(3)=1
  CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)
  CALL MESS(5, RC(13,1), RC(13,2), RC(13,3), 7)
  CLOSE (SUNIT1)
  GOTO 5

```

```

C CHECK TO SEE IF THE SOURCE IS IN THE DATA BASE

```

```

25  OPEN (SUNIT2, FILE=SFILE2, STATUS='OLD', RECL=40, ACCESS='DIRECT',
*    FORM='UNFORMATTED')
32  DO 30 II=1,9,1
    IF (SINIT1(II) .EQ. 0) GOTO 30
    READ(SUNIT2, REC=SINIT1(II)) CINIT2
    IF (MKEY(2) .EQ. CINIT2) GOTO 35
30  CONTINUE
    IF (SINIT1(10) .NE. 0) THEN
        I=SINIT1(10)
        READ(SUNIT1, REC=I) CINIT1, SINIT1, S2
        GOTO 32
                                ENDIF

    IST=2
    ST(3)=1
    INP(1)=' '
    CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)
    CALL MESS(5, RC(13, 1), RC(13, 2), RC(13, 3), 7)
    CLOSE (SUNIT1)
    CLOSE (SUNIT2)
    GOTO 5

35  OPEN (SUNIT3, FILE=SFILE3, STATUS='OLD', RECL=40, ACCESS='DIRECT',
*    FORM='UNFORMATTED')
    OPEN (SUNIT4, FILE=SFILE4, STATUS='OLD', RECL=16,
*    ACCESS='DIRECT', FORM='UNFORMATTED')
    READ(SUNIT3, REC=1) HD3
    READ(SUNIT4, REC=HD3(II)) S3

C   CONVERT THE BINARY DATA TO ALPHA DATA
    DO 40 K=3,7,1
        WRITE(TEMP(K) (1:8), '(F8.3)') S2(K-2)
40  CONTINUE
    DO 41 K=8,11,1
        WRITE(TEMP(K) (1:8), '(F8.3)') S3(K-7)
41  CONTINUE

C   BLANK OUT THE COMMAND LINE
    ST(3)=1
    INP(1)=' '
    CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)

C   OUTPUT THE STORED DATA
    ST(3)=0
    CALL MENUWR(RC, 13, 3, 11, TEMP, 0, 1, ST)

    IST=3
55  CALL MENURD(RC, 13, IST, 11, TEMP, ITR)

C   SCANNING THE COMMAND INPUT ROW
60  INP(1)=' '
    CALL MENURD(RC, 13, 12, 12, INP, ITT)
    IF (INP(1) .EQ. ' ') THEN
        IST=3
        ST(3)=1
        CALL MESS(4, RC(13, 1), RC(13, 2), RC(13, 3), 1)
        CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)
        GOTO 55
    ENDIF
    IF (INP(1) .EQ. 'X') THEN
        CLOSE (SUNIT1)

```

```

                                CLOSE (SUNIT2)
                                CLOSE (SUNIT3)
                                CLOSE (SUNIT4)
                                OPT='&'
                                GOTO 100
                                ENDIF
                                IF (INP(1) .NE. 'M') GOTO 60

C    CHECK THE REQUEST FOR ERRORS
    TEMP(1)(1:8)=' '
    TEMP(2)(1:8)=' '
    CALL MENUCK(TEMP, TDATA, 11, ' (F8.3)', IERR)
    IF (IERR. NE. 0) THEN
        INP(1)=' '
        ST(3)=1
        CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)
        CALL MESS(1, RC(13, 1), RC(13, 2), RC(13, 3), 7)
        IST=IERR
        GOTO 55
    ENDIF

DO 61 K=3, 6, 1
    IF (TDATA(K) .LE. 0.0) THEN
        IST=K
        INP(1)=' '
        ST(3)=1
        CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)
        CALL MESS(1, RC(13, 1), RC(13, 2), RC(13, 3), 7)
        GOTO 55
    ENDIF

61  CONTINUE
    IF (TDATA(8) .LE. 0) THEN
        IST=8
        INP(1)=' '
        ST(3)=1
        CALL MENUWR(RC, 13, 12, 12, INP, 0, 1, ST)
        CALL MESS(1, RC(13, 1), RC(13, 2), RC(13, 3), 7)
        GOTO 55
    ENDIF

C    LOAD THE S2 AND S3 ARRAYS WITH THE NEW DATA
    S3(1)=TDATA(8)
    S3(2)=TDATA(9)
    S3(3)=TDATA(10)
    S3(4)=TDATA(11)
    DO 42 K=1, 10, 1
        S2(K)=0.0
42  CONTINUE
    DO 43 K=1, 5, 1
        S2(K)=TDATA(K+2)
43  CONTINUE

    WRITE(SUNIT4, REC=HD3(II)) S3
    DO 44 I=2, HD, 1
        READ(SUNIT1, REC=I) CINIT1, SINIT1
        IF (CINIT1 .EQ. MKEY(1)) GOTO 45
44  CONTINUE
45  WRITE(SUNIT1, REC=I) CINIT1, SINIT1, S2
80  IF (SINIT1(10) .NE. 0) THEN
        I=SINIT1(10)
        READ(SUNIT1, REC=I) CINIT1, SINIT1

```

```

WRITE(SUNIT1, REC=1) CINIT1, SINIT1, S2
GOTO 80

                                ENDIF

CLOSE (SUNIT1)
CLOSE (SUNIT2)
CLOSE (SUNIT3)
CLOSE (SUNIT4)
OPT='&'
GOTO 100

END
SUBROUTINE SBQST(SKEY, SUNIT1, SUNIT2, SFILE1, SFILE2, SMUNIT, SMFILE,
*               ITR, EFLAG)

CHARACTER*70  OUT(19)
CHARACTER*40  SKEY(1), HEAD, THEAD, SHEAD
CHARACTER*7   SFILE1, SFILE2, SMFILE
CHARACTER*3   CMD(1)

INTEGER       SUNIT1, SUNIT2, SMUNIT, RC(19, 3), ST(3), ICNT, ITR,
*             EFLAG, HD, HDR(10)

LOGICAL       FLAG1

DATA ST/0, 0, 0/

EFLAG=1
OPEN (SUNIT1, FILE=SFILE1, STATUS='OLD', RECL=120, ACCESS='DIRECT',
*     FORM='UNFORMATTED')
OPEN (SUNIT2, FILE=SFILE2, STATUS='OLD', RECL=40, ACCESS='DIRECT',
*     FORM='UNFORMATTED')
5 READ(SUNIT1, REC=1) HD
   ICNT=2
   FLAG1=.FALSE.
   DO 20 I=2, HD, 1
     IF (.NOT. FLAG1) FLAG1=.TRUE.
     READ(SUNIT1, REC=I) HEAD, HDR
     ICHECK=0
     DO 13 JG=1, 9, 1
       IF (HDR(JG) .NE. 0) ICHECK=1
13    CONTINUE
     IF (ICHECK .EQ. 0) GOTO 20
     IF (SKEY(2) .NE. '*') THEN
     IF (SKEY(2) .NE. '?') THEN
       DO 12 K=1, 9, 1
         IF (HDR(K) .EQ. 0) GOTO 12
         READ(SUNIT2, REC=III) SHEAD
         IF (SHEAD .EQ. SKEY(2)) GOTO 11
12    CONTINUE
       GOTO 20
                                ENDIF
                                ENDIF

       DO 10 J=2, I-1, 1
         READ(SUNIT1, REC=J) THEAD
         IF (THEAD .EQ. HEAD) GOTO 20
10    CONTINUE
11    EFLAG=0
       OUT(ICNT)(1:70)=' '
       WRITE(OUT(ICNT)(1:3), '(I3)') I
       OUT(ICNT)(6:70)=HEAD

```

```

15      IF (ICNT .EQ. 19) THEN
          CALL MENUHV(SMFILE,182,RC,19,SMUNIT)
          CALL MENUWR(RC,19,2,19,OUT,0,1,ST)
          CMD(1)=' '
          CALL MENURD(RC,19,1,1,CMD,ITR)
          IF (CMD(1) .EQ. 'X ') THEN
              CLOSE (SUNIT1)
              CLOSE (SUNIT2)
              RETURN
          ENDIF
          IF (CMD(1) .EQ. 'C ') THEN
              ICNT=2
              FLAG1=.FALSE.
              GOTO 20
          ENDIF
          READ(CMD(1)(1:3),'(I3)',ERR=15) II
          IF ((II .LT. 2).OR.(II .GT. HD)) GOTO 15
          READ(SUNIT1,REC=II) SKEY(1)
          CLOSE (SUNIT1)
          CLOSE (SUNIT2)
          RETURN
      ENDIF

20      ICNT=ICNT+1
      CONTINUE
      IF (EFLAG .NE. 0) RETURN
      IF (.NOT. FLAG1) GOTO 5
      CALL MENUHV(SMFILE,182,RC,19,SMUNIT)
      CALL MENUWR(RC,19,2,ICNT-1,OUT,0,1,ST)
25      CMD(1)=' '
      CALL MENURD(RC,19,1,1,CMD,ITR)
      IF (CMD(1) .EQ. 'X ') THEN
          CLOSE (SUNIT1)
          CLOSE (SUNIT2)
          RETURN
      ENDIF
      IF (CMD(1) .EQ. 'C ') THEN
          FLAG1=.FALSE.
          GOTO 5
      ENDIF
      READ(CMD(1)(1:3),'(I3)',ERR=25) II
      IF ((II .LT. 2).OR.(II .GT. HD)) GOTO 25
      READ(SUNIT1,REC=II) SKEY(1)
      CLOSE (SUNIT1)
      CLOSE (SUNIT2)
      RETURN

END
SUBROUTINE SRCQST(SKEY,SUNIT1,SUNIT2,SFILE1,SFILE2,SMUNIT,SMFILE,
*                ITR,EFLAG)

CHARACTER*70  OUT(19)
CHARACTER*40  SKEY(1),HEAD,THEAD,SHEAD
CHARACTER*7   SFILE1,SFILE2,SMFILE
CHARACTER*3   CMD(1)

INTEGER       SUNIT1,SUNIT2,SMUNIT,RC(19,3),ST(3),ICNT,ITR,
*             EFLAG,HD,HD1

INTEGER       HDR(10)

```

```

LOGICAL      FLAG1

DATA ST/0,0,0/

EFLAG=1
OPEN (SUNIT1,FILE=SFILE1,STATUS='OLD',RECL=120,ACCESS='DIRECT',
*   FORM='UNFORMATTED')
OPEN (SUNIT2,FILE=SFILE2,STATUS='OLD',RECL=40,ACCESS='DIRECT',
*   FORM='UNFORMATTED')
READ(SUNIT2,REC=1) HD
5  ICNT=2
   FLAG1=.FALSE.
   DO 20 I=2,HD,1
     IF (.NOT. FLAG1) FLAG1=.TRUE.
     READ(SUNIT2,REC=I) SHEAD
     IF (SKEY(1) .NE. '*') THEN
       IF (SKEY(1) .NE. '?') THEN
         READ(SUNIT1,REC=1) HD1
         DO 12 K=2,HD1,1
           READ(SUNIT1,REC=K) HEAD,HDR
           IF (HEAD .NE. SKEY(1)) GOTO 12
           DO 13 KK=1,9,1
             IF (HDR(KK) .EQ. I) GOTO 11
13          CONTINUE
12          CONTINUE
           GOTO 20
         ENDIF
         ENDIF
11         EFLAG=0
         OUT(ICNT)(1:70)=' '
         WRITE(OUT(ICNT)(1:3),'(I3)') I
         OUT(ICNT)(6:70)=SHEAD
         IF (ICNT .EQ. 19) THEN
           CALL MENUSV(SMFILE,183,RC,19,SMUNIT)
           CALL MENUWR(RC,19,2,19,OUT,0,1,ST)
15          CMD(1)=' '
           CALL MENURD(RC,19,1,1,CMD,ITR)
           IF (CMD(1) .EQ. 'X ') THEN
             CLOSE (SUNIT1)
             CLOSE (SUNIT2)
             RETURN
             ENDIF
           IF (CMD(1) .EQ. 'C ') THEN
             ICNT=2
             FLAG1=.FALSE.
             GOTO 20
             ENDIF
           READ(CMD(1)(1:3),'(I3)',ERR=15) II
           IF ((II .LT. 2).OR.(II .GT. HD)) GOTO 15
           READ(SUNIT2,REC=II) SKEY(2)
           CLOSE (SUNIT1)
           CLOSE (SUNIT2)
           RETURN
             ENDIF
20          ICNT=ICNT+1
           CONTINUE
           IF (EFLAG .NE. 0) RETURN
           IF (.NOT. FLAG1) GOTO 5
           CALL MENUSV(SMFILE,183,RC,19,SMUNIT)
           CALL MENUWR(RC,19,2,ICNT-1,OUT,0,1,ST)

```

```

25  CMD(1)= ' '
    CALL MENURD(RC,19,1,1,CMD,ITR)
    IF (CMD(1) .EQ. 'X ' ) THEN
        CLOSE (SUNIT1)
        CLOSE (SUNIT2)
        RETURN
    ENDIF
    IF (CMD(1) .EQ. 'C ' ) THEN
        FLAG1=.FALSE.
        GOTO 5
    ENDIF
    READ(CMD(1)(1:3),'(I3)',ERR=25) II
    IF ((II .LT. 2).OR.(II .GT. HD)) GOTO 25
    READ(SUNIT2,REC=II) SKEY(2)
    CLOSE (SUNIT1)
    CLOSE (SUNIT2)
    RETURN
END

```